



Linguagens de Programação

Aula 15

Celso Olivete Júnior

`olivete@fct.unesp.br`

Na aula passada

- Linguagem *Haskell*

Na aula de hoje

- ❑ Paradigma Lógico – Linguagem *Prolog*
(*PRO*grammation en *LOG*ique)

Introdução

- ❑ Em linguagens de programação (LPs) **imperativas** um programa é composto por uma **sequência** de **comandos** que representam as atividades computacionais que serão executadas.
- ❑ O programador deve portanto **especificar** claramente como realizar o processamento desejado, ou seja, como é o **algoritmo**.

Introdução

- ❑ Em LPs lógicas um programa consiste na definição de relações lógicas que devem ser satisfeitas pela solução procurada.
- ❑ A busca de uma solução ocorre automaticamente através de *regras de inferência*.

Introdução

- ❑ Programar em uma LP lógica consiste em:
 - Declarar **fatos** primitivos sobre um domínio
 - Definir **regras** que expressam relações entre os fatos do domínio
 - Fazer **perguntas** sobre o domínio

Exemplo de programação em lógica

Sócrates é homem.
Todo homem é mortal.

Quem é mortal?

Sócrates é mortal.



homem(sócrates).
mortal(X) ← homem(X).

?- mortal(Z).

Z = sócrates.

Fatos, regras e consultas

□ A programação em lógica baseia-se em estruturas lógicas denominadas **Cláusulas de Horn**, que se apresentam em quatro formas distintas:

- Fatos: $a \leftarrow$ Verdades incondicionais
- Regras $a \leftarrow b$ Podem ou não serem verdadeiras
- Consultas: $\leftarrow b$ Provocam a execução do programa
- Vazia: \leftarrow

Fatos , regras e consultas

❑ Fatos → São verdades incondicionais:

`pai(josé, joão). //josé é pai de joão?`

`pai(joão, júlio).`

`pai(júlio, jorge).`

Fatos , regras e consultas

- ❑ Fatos → São verdades incondicionais:

`pai(josé, joão). //josé é pai de joão?`

`pai(joão, júlio).`

`pai(júlio, jorge).`

- ❑ Regras → Podem ser verdadeiras ou não:

`filho(X, Y) ← pai(Y, X).`

`avô(X, Y) ← pai(X, Z), pai(Z, Y).`

Fatos , regras e consultas

- ❑ Fatos → São verdades incondicionais:

pai(josé, joão). //josé é pai de joão?
pai(joão, júlio).
pai(júlio, jorge).

- ❑ Regras → Podem ser verdadeiras ou não:

filho(X, Y) ← pai(Y, X).
avô(X, Y) ← pai(X, Z), pai(Z, Y).

- ❑ Consultas → Provocam a execução do programa:

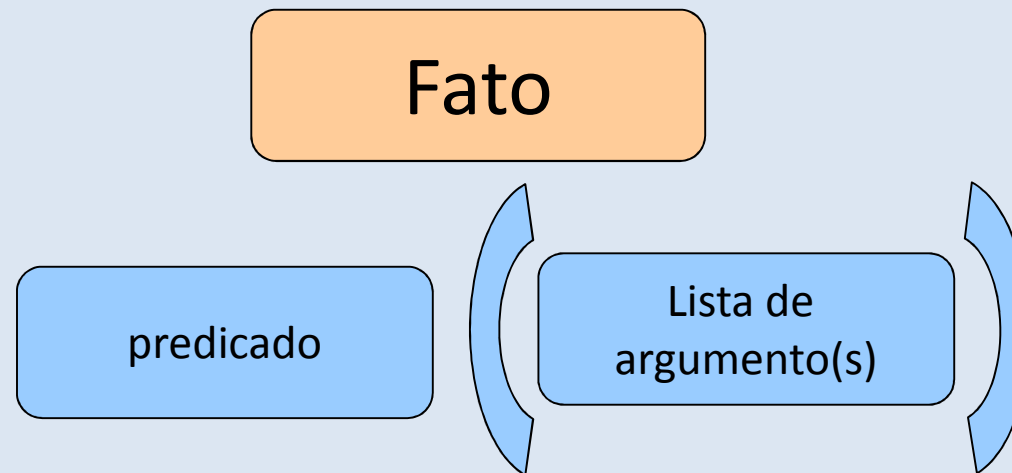
?- pai(júlio, X). X=jorge
?- filho(júlio, X). X=joão

Fatos , regras e consultas

- ❑ Fatos → declaração

Uma forma de declarar um fato como “**uma baleia é um mamífero**” é:

mamífero(baleia)



Fatos

exemplos

- ❑ Para representar o fato “Bruno gosta de Ana”

`gostar(bruno, ana)`

predicado

- ❑ Para representar “Ana gosta de Bruno”

`gostar(ana, bruno)`

argumentos

- ❑ Nas expressões acima gostar é predicado do fato, representando uma relação entre os argumentos

Fatos

exemplos

- ❑ Para representar o fato "Bruno gosta de Ana"

`gostar(bruno, ana)`

predicado

- ❑ Note que este fato é diferente de "Ana gosta de Bruno"

`gostar(ana, bruno)`

argumentos

Como
relacionar
fatos?

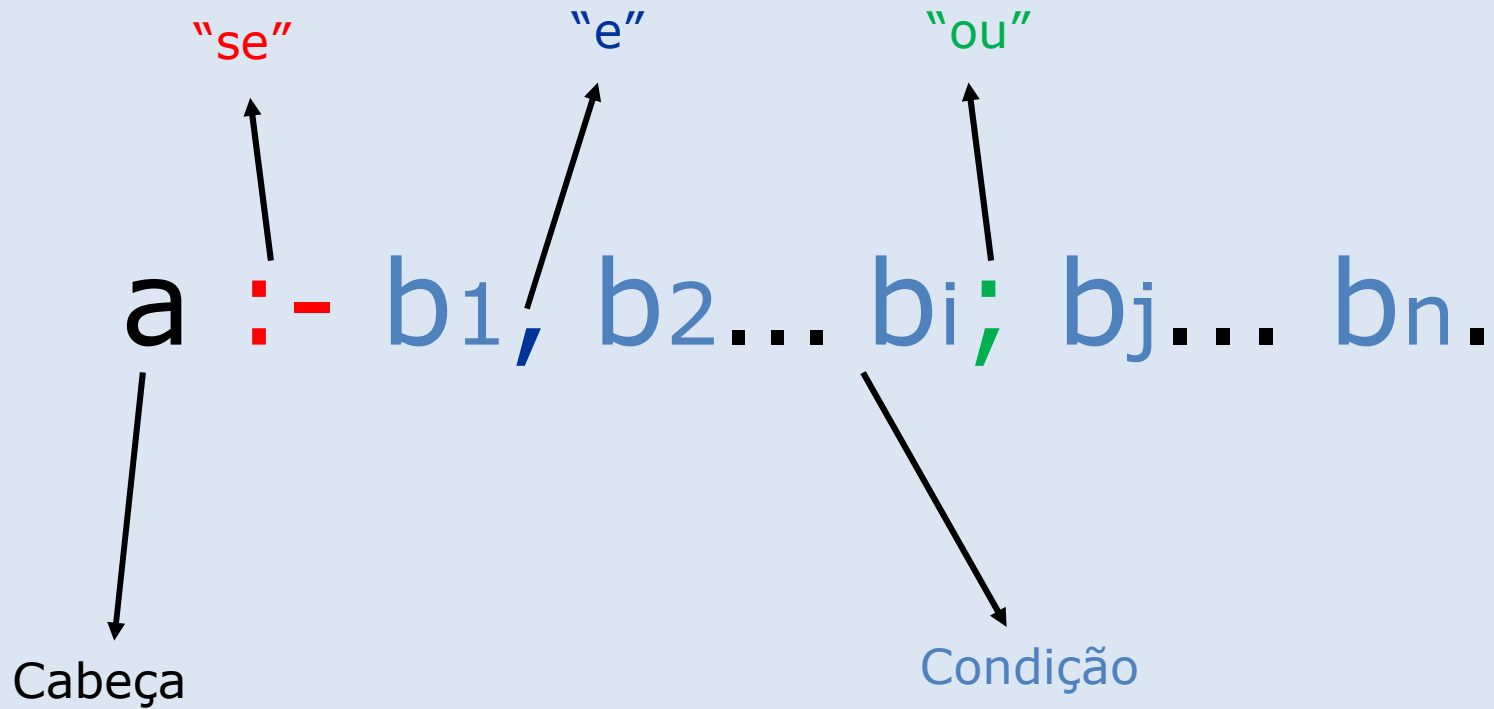
- ❑ Nas expressões acima gostar é predicado do fato, representando uma relação entre os argumentos

Regras

- ❑ Fatos são relacionados através de **Regras**
 - ❑ denominadas **Cláusulas de Horn**

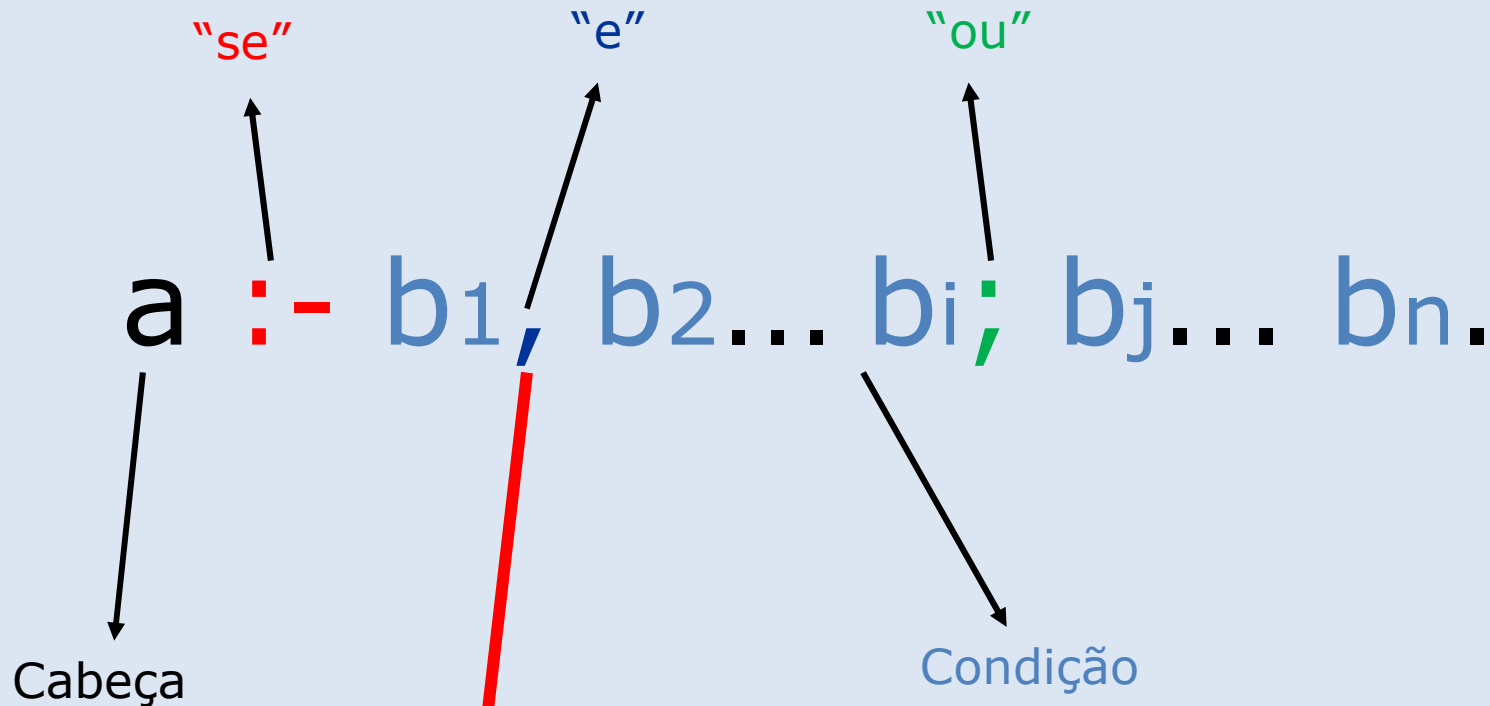
Regras

A cláusula *Prolog*



Regras

A cláusula *Prolog*



se os predicados de 1 a n são verdadeiros para as variáveis b_i , então o predicado **pred** é verdadeiro para as variáveis b_i

Regras exemplo

- ❑ Fatos → São verdades incondicionais:

```
pai(josé, joão). //josé é pai de joão?  
pai(joão, júlio).  
pai(júlio, jorge).
```

- ❑ Regras → Podem ser verdadeiras ou não:

```
filho(X, Y):- pai(Y, X).  
avô(X, Y) :- pai(X, Z), pai(Z, Y).
```

Cláusula “**se**”

Cláusula “**e**”

Operadores *Prolog*

Linguagem Natural	Programas Prolog
E	,
OU	;
SE	:-
NÃO	not

Consultas

- ❑ Consultas provocam a execução do programa

- ❑ **Fatos**

pai(josé, joão). //josé é pai de joão?
pai(joão, júlio).
pai(júlio, jorge).

- ❑ **Consultas**

?- pai(júlio, X).
?- pai(X, Y).

X unificou a jorge

X=jorge
X=joão Y = júlio
X=josé Y = joão
X=júlio Y = jorge

X unificou ao primeiro argumento
Y unificou ao segundo argumento

Consultas

mais exemplos

- ❑ Fatos

animal (cachorro)

animal (gato)

- ❑ Consultas

?- animal(cachorro)

yes

?- animal(X)

cachorro;

gato;

Unificação

- ❑ Para tentar provar um fato, **Prolog** precisa estabelecer equivalência entre fatos

equivalentes = unificáveis

- ❑ Dois átomos são unificáveis apenas se são idênticos
- ❑ Duas estruturas são idênticas se o seu predicado for idêntico, e se seus argumentos forem unificáveis

Unificação

- ❑ Uma variável é unificável a qualquer coisa
- ❑ Durante o processo de resolução, uma variável é instanciada com o valor que permite a sua unificação com um símbolo correspondente de outro fato
- ❑ A unificação é representada por =

Unificação

Exemplos

fruta(manga).

?- fruta(X). → *perguntando...*

X = manga → X foi unificado com "manga"

gostar(bruno,ana).

?- gostar(bruno,X). → *perguntando...*

X = ana → X foi unificada com "ana"

?- X=sol

X =sol → X foi unificada com "sol"

?- sol = sol

yes → "sol" foi unificado com "sol"

Unificação

Exemplos

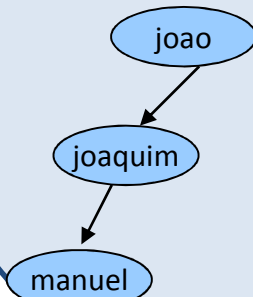
Termo 1	Termo 2	Resultado da unificação
henrique	henrique	unificam
eduardo	henrique	não unificam
Y	8.5	Y=8.5
data(25,maio,Ano)	data(D,maio,1981)	D=15 Ano=1981
pai_de(X, eduardo)	pai_de(henrique,Y)	X=henrique Y=eduardo

Exemplo de processo de resolução

pai(joao, joaquim).
pai(joaquim, manuel).
avo(X,Y):- pai(Z,Y), pai(X,Z).

?- avo(joao,Z).
X = manuel

?- avo(joao,manuel).
yes.



avo(X,Y):- pai(Z,Y), pai(X,Z).

Processo de resolução

avo(joao,Z):- pai(joao,joaquim), pai(joaquim,manuel).

No fato pai(Z,Y) a variável Z foi unificada a “joao” e a variável Y foi unificada a “joaquim”

No fato pai(X,Z) a variável X foi unificada a “joaquim” e a variável Z foi unificada a “manuel”

avo(X,Y):- pai(Z,Y), pai(X,Z).

Processo de resolução

avo(joao,manuel):- pai(joaquim,manuel), pai(joao,joaquim).

No fato pai(Z,Y) a variável Z foi unificada a “joaquim” e a variável Y foi unificada a “manuel”

No fato pai(X,Z) a variável X foi unificada a “joao” e a variável Z foi unificada a “joaquim”

Declaração de fatos

- ❑ Os fatos na linguagem Prolog são representados através de **átomos** ou estruturas.
- ❑ **Átomos** são strings que **começam** sempre com **letra minúscula**. Exemplos:
 - esta_frio
 - teste
 - dinheiro

Declaração de fatos

- ❑ Estruturas são átomos seguidos de uma lista de argumentos entre parênteses:

pred (arg1, arg2, argN).

onde:

- **pred** é o nome do predicado
- **argi** são os argumentos
- **N** é a aridade (número de argumentos)
- **.** representa o final de uma cláusula

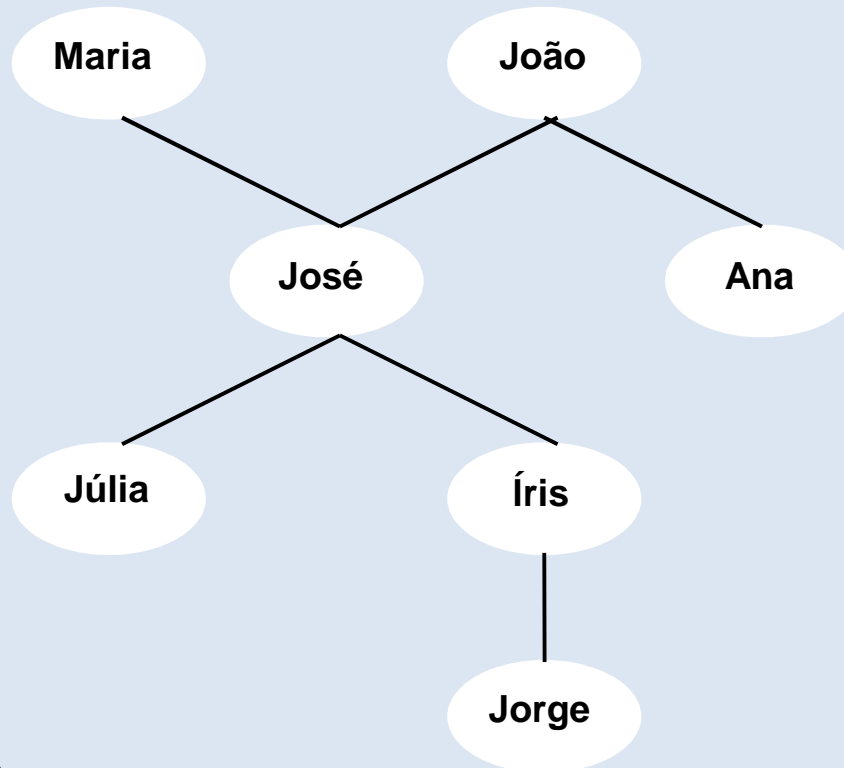
Declaração de fatos

- ❑ Os argumentos de uma estrutura podem ser átomos, estruturas, variáveis e números.
- ❑ Exemplos:
 - carro**(preto)
 - amigo**(joana,pedro)
 - arvore_bin**(valor1, arvore_bin(valor2, valor3))
- ❑ Estruturas representam relações entre os argumentos, ou seus predicados

Fatos

mais exemplos

- Seja a árvore genealógica mostrada abaixo...



que pode ser **representada** pelo seguinte **programa** Prolog:

```
progenitor(maria, josé).  
progenitor(joão, josé).  
progenitor(joão, ana).  
progenitor(josé, júlia).  
progenitor(josé, íris).  
progenitor(íris, jorge).
```

Este programa representa a relação **progenitor**, na forma de um **predicado**, que contém 6 **cláusulas**, que são todas **fatos**.

Consultas

❑ O programa pode ser pensado como uma tabela em um BD.

progenitor(maria, josé).

progenitor(joão, josé).

progenitor(joão, ana).

progenitor(josé, júlia).

progenitor(josé, íris).

progenitor(íris, jorge).

No caso de um programa constituído unicamente de fatos, a semântica é exatamente a mesma de uma BD relacional...

que pode ser consultada de várias maneiras:

?- **progenitor**(joão, ana).

yes

?- **progenitor**(joão, jorge).

no

?- **progenitor**(joão, X).

X=josé;

X=ana;

no

?- **progenitor**(X,Y).

X=maria, Y=josé;

X=joão, Y=josé;

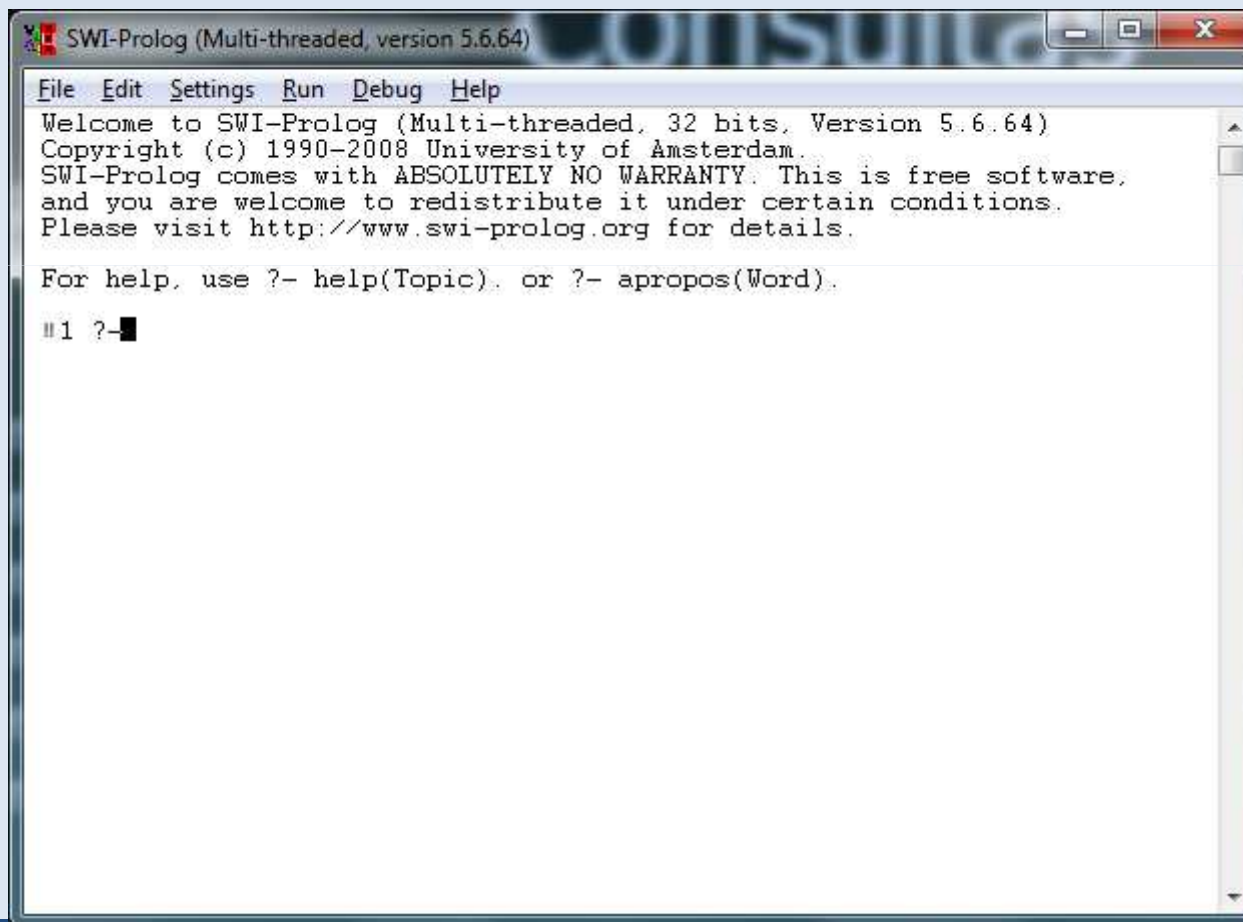
...

X=íris, Y=jorge;

no

Definindo *fat*os

❑ O ambiente *SWI-Prolog*



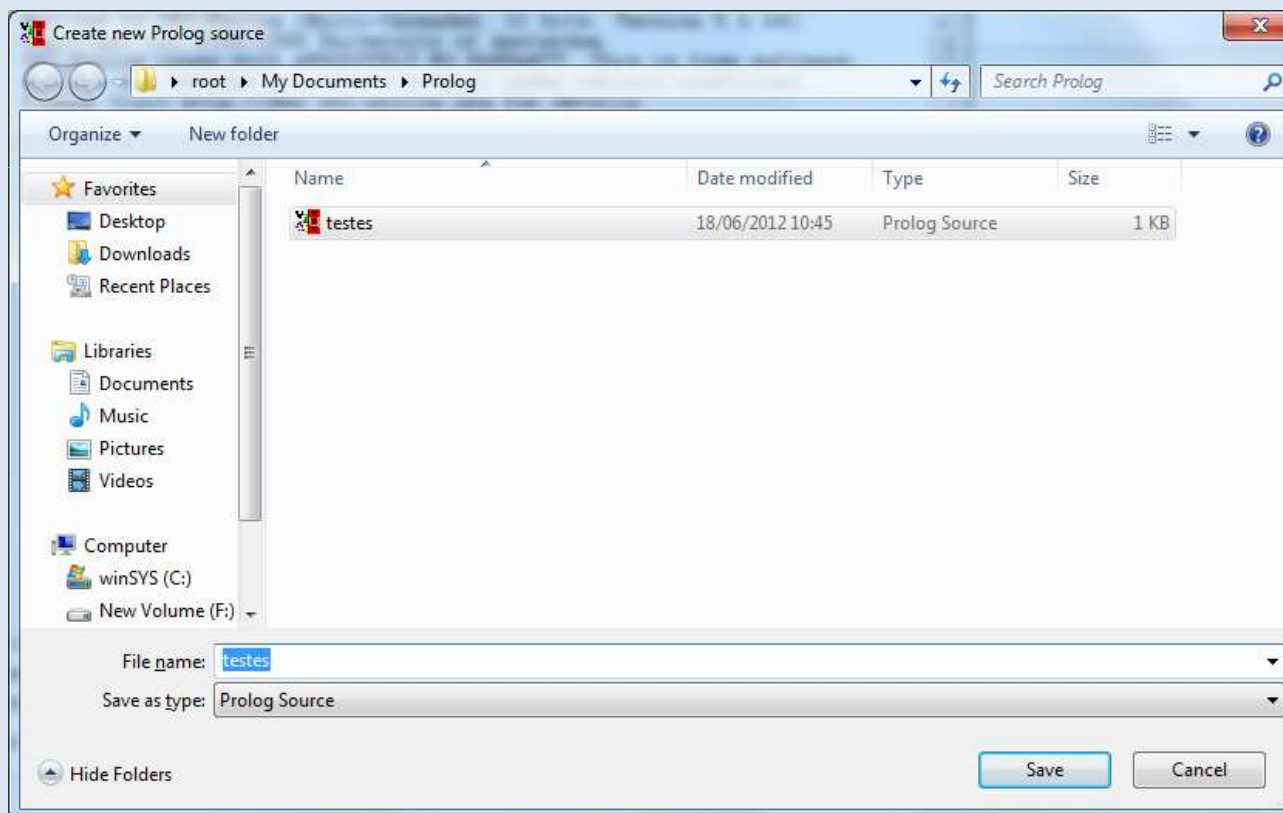
```
SWI-Prolog (Multi-threaded, version 5.6.64)
File Edit Settings Run Debug Help
Welcome to SWI-Prolog (Multi-threaded, 32 bits, Version 5.6.64)
Copyright (c) 1990-2008 University of Amsterdam.
SWI-Prolog comes with ABSOLUTELY NO WARRANTY. This is free software,
and you are welcome to redistribute it under certain conditions.
Please visit http://www.swi-prolog.org for details.

For help, use ?- help(Topic), or ?- apropos(Word).

#1 ?-
```


Definindo *fat*os

- ❑ Definindo um programa: File/New...



Definindo *fatos*

- ❑ Defina os fatos e salve o arquivo

`progenitor(maria, josé).`

`progenitor(joão, josé).`

`progenitor(joão, ana).`

`progenitor(josé, júlia).`

`progenitor(josé, íris).`

`progenitor(íris, jorge).`

Prolog é *case sensitive*!!

O ambiente SWI-Prolog

❑ Para “carregar” o arquivo do programa
?-**consult**(`nome_arquivo.extensão`)

Ou

File/Consult...

❑ Para visualizar o conhecimento existente na
base de dados

listing.

Consultas

❑ O programa pode ser pensado como uma tabela em um BD.

progenitor(maria, josé).

progenitor(joão, josé).

progenitor(joão, ana).

progenitor(josé, júlia).

progenitor(josé, íris).

progenitor(íris, jorge).

No caso de um programa constituído unicamente de fatos, a semântica é exatamente a mesma de uma BD relacional...

que pode ser consultada de várias maneiras:

?- progenitor(joão, ana).

yes

?- progenitor(joão, jorge).

no

?- progenitor(joão, X).

X=josé;

X=ana;

no

?- progenitor(X,Y).

X=maria, Y=josé;

X=joão, Y=josé;

...

X=íris, Y=jorge;

no

Para que mais de uma associação possa ser encontrada, basta pressionar a tecla ; após cada resultado apresentado.

Ampliando a base de fatos

O programa pode ser ampliado acrescentando-se novos fatos que inclusive podem estabelecer novas relações.

No exemplo ao lado acrescentou-se ao programa original as relações **masculino** e **feminino**.

Estas relações, que possuem **aridade 1**, podem ser pensadas como sendo **atributos** dos objetos a que se aplicam.

progenitor(maria, José).
progenitor(joão, José).
progenitor(joão, ana).
progenitor(josé, júlia).
progenitor(josé, íris).
progenitor(íris, jorge).

masculino(joão).
masculino(josé).
masculino(jorge).

feminino(maria).
feminino(ana).
feminino(júlia).
feminino(íris).

Regras

As relações **pai** e **mãe** podem agora ser definidas da seguinte maneira:

X é pai de Y se X é **progenitor** de Y e X é **masculino**.

X é mãe de Y se X é **progenitor** de Y e X é **feminino**.

Em Prolog:

X é pai de Y

`pai(X,Y) :-`

`progenitor(X,Y),`
`masculino(X).`

`mãe(X,Y) :-`

`progenitor(X,Y),`
`feminino(X).`

e (,)

Mais regras

As relações **irmão** e **irmã** podem agora ser definidas da seguinte maneira:

X é irmão de Y **se**
Z é **progenitor** de X **e**
Z é **progenitor** de Y **e**
X é **masculino**.

X é irmã de Y **se** ...

Em Prolog:

```
irmão(X,Y) :-  
    progenitor(Z,X),  
    progenitor(Z,Y),  
    masculino(X).
```

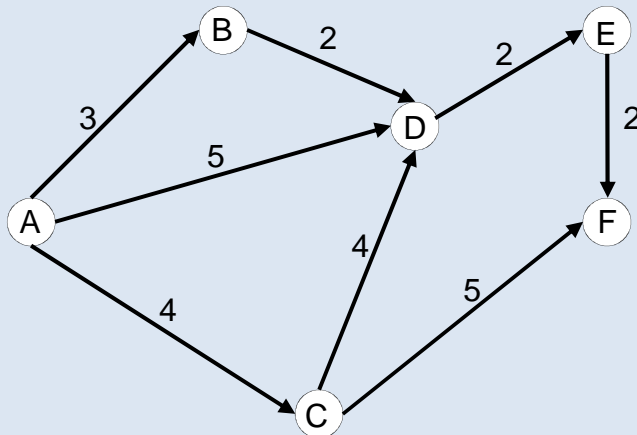
```
irmã(X,Y) :- ...'''
```

Exercício

- ❑ Acrescentar ao programa da árvore genealógica as seguintes relações
 - avô
 - tia
 - prima

Exemplo: caminhos em grafos

□ Dado o grafo direcionado



□ E as relações $conecta(X,Y,K)$ e $caminho(X,Y,K)$, onde X e Y são nodos e K é o custo entre eles. Ao lado, em Prolog:

```

conecta(a,b,3).
conecta(a,c,4).
...
conecta(e,f,2).
  
```

```

caminho(X,Y,K):-
    conecta(X,Y,K).
caminho(X,Y,K):-
    conecta(X,Z,K1),
    caminho(Z,Y,K2),
    K is K1+K2.
  
```

```

?- caminho(X,Y,K).
X=a, Y=b, K=3;
...
no
  
```

Fluxo de controle no processo de resolução

- ❑ O Prolog utiliza um fluxo de controle implícito no processo de resolução de uma questão
- ❑ O sentido deste fluxo é determinístico → segue sempre a mesma ordem
 - ❑ O processo de resolução segue a ordem em que os fatos/regras foram declarados
 - ❑ Para cada predicado correto encontrado, os argumentos são testados em ordem
 - ❑ Para cada regra, as condições são testadas sempre da esquerda para direita.

Fluxo de controle

Exemplo

tropical(caribe).

tropical(havai).

praia(caribe).

praia(havai).

bonito(havai).

bonito(caribe).

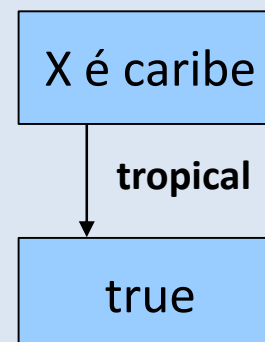
paraiso_tropical(X) :- tropical(X), praia(X), bonito(X).

Questionando...

?- paraiso_tropical(X).

X = caribe

- O processo de resolução segue a ordem em que os fatos/regras foram declarados
- Para cada predicado correto encontrado, os argumentos são testados em ordem
- Para cada regra, as condições são testadas sempre da esquerda para direita.



Fluxo de controle

Exemplo

tropical(caribe).

tropical(havai).

praia(caribe).

praia(havai).

bonito(havai).

bonito(caribe).

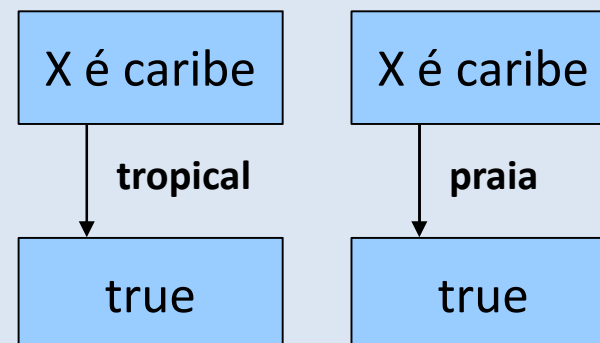
paraiso_tropical(X) :- tropical(X), praia(X), bonito(X).

Questionando...

?- paraiso_tropical(X).

X = caribe

- O processo de resolução segue a ordem em que os fatos/regras foram declarados
- Para cada predicado correto encontrado, os argumentos são testados em ordem
- Para cada regra, as condições são testadas sempre da esquerda para direita.



Fluxo de controle

Exemplo

tropical(caribe).
 tropical(havai).
 praia(caribe).
 praia(havai).
 bonito(havai).
 bonito(caribe).

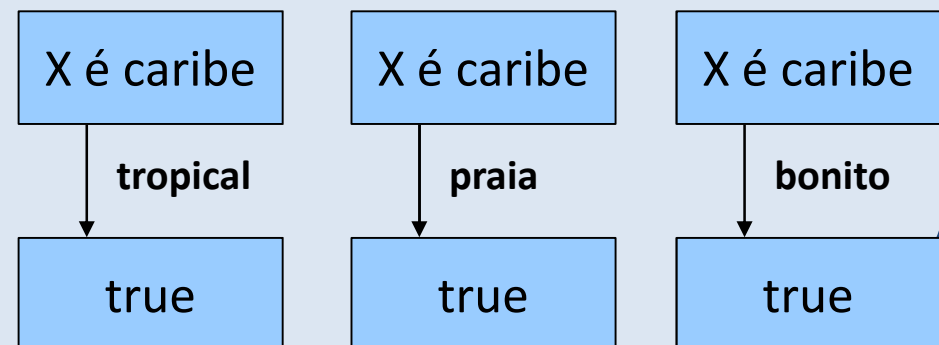
paraiso_tropical(X) :- tropical(X), praia(X), bonito(X).

Questionando...

?- paraiso_tropical(X).

X = caribe

- O processo de resolução segue a ordem em que os fatos/regras foram declarados
- Para cada predicado correto encontrado, os argumentos são testados em ordem
- Para cada regra, as condições são testadas sempre da esquerda para direita.



Fluxo de controle

- Variáveis são instanciadas implicitamente com valores que permitem a **unificação** da estrutura
- Assim que uma associação válida for encontrada, os valores com os quais as variáveis foram instanciadas são impressos
- Apenas a primeira associação válida é impressa!!*
- Para que mais de uma associação possa ser encontrada, basta pressionar a tecla ; após cada resultado apresentado.

Fluxo de controle implícito

- ❑ Para provar questões mais complexas, Prolog pode ser obrigado a testar várias vezes a mesma condição, instanciando uma variável com diferentes valores

Esse processo de retornar e testar novamente uma condição é denominado ***backtracking***

Fluxo de controle implícito

- **Exemplo**

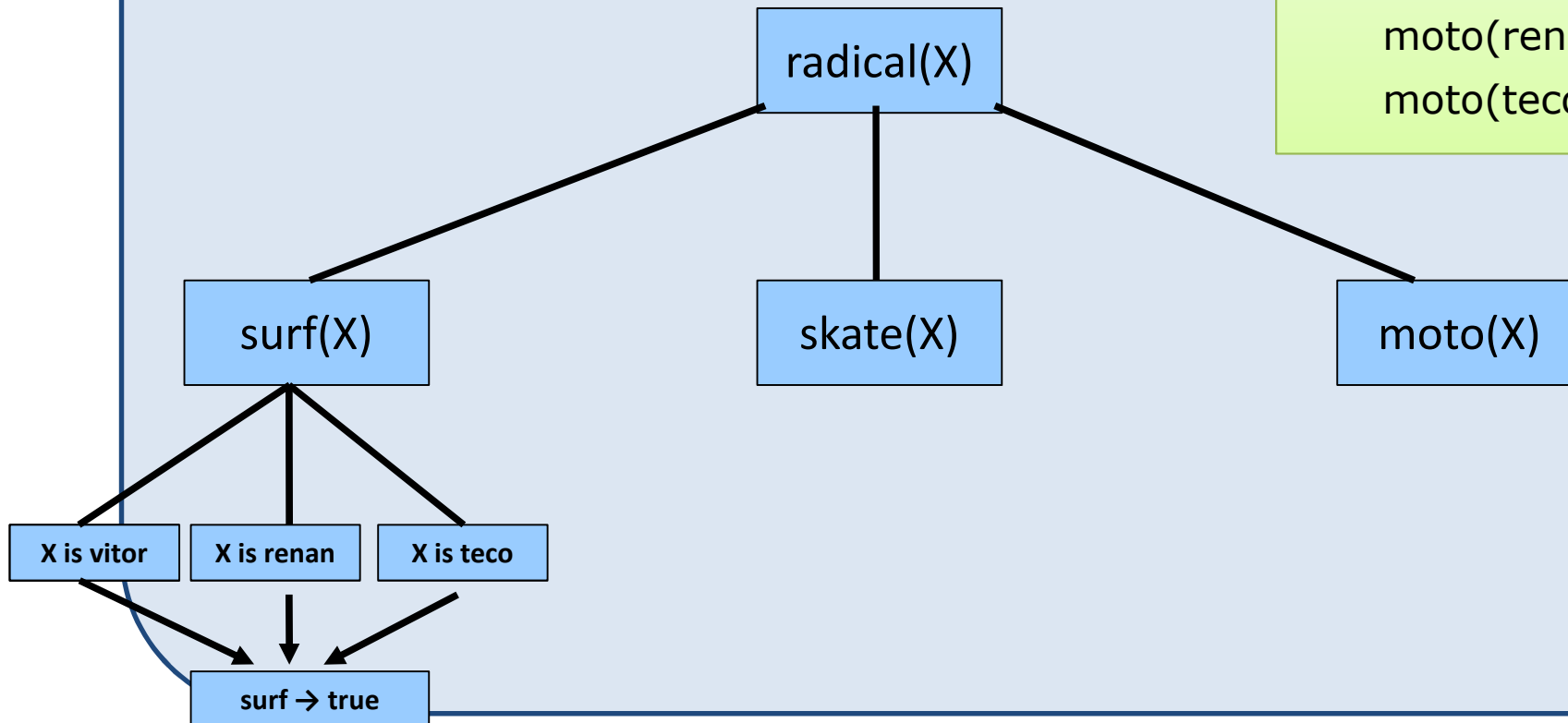
```
surf(vitor).  
surf(renan).  
surf(teco).  
skate(vitor).  
skate(teco).  
moto(renan).  
moto(teco).  
radical(X):- surf(X), skate(X), moto(X).  
?- X is teco
```


Fluxo de controle implícito

❑ Fluxo do percurso:

`radical(X):- surf(X), skate(X), moto(X).`

```
surf(vitor).
surf(renan).
surf(teco).
skate(vitor).
skate(teco).
moto(renan).
moto(teco).
```

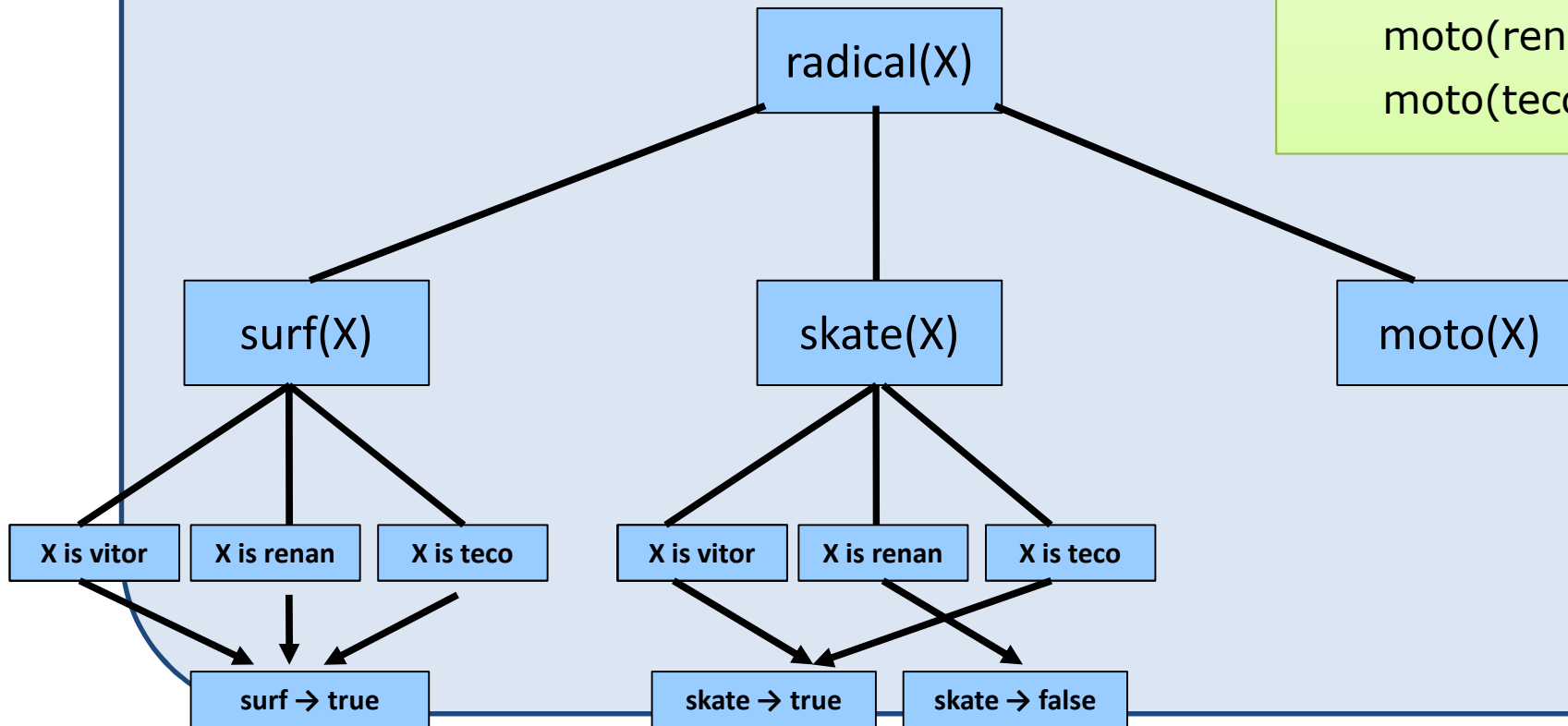


Fluxo de controle implícito

❑ Fluxo do percurso:

`radical(X):- surf(X), skate(X), moto(X).`

surf(vitor).
 surf(renan).
 surf(teco).
 skate(vitor).
 skate(teco).
 moto(renan).
 moto(teco).

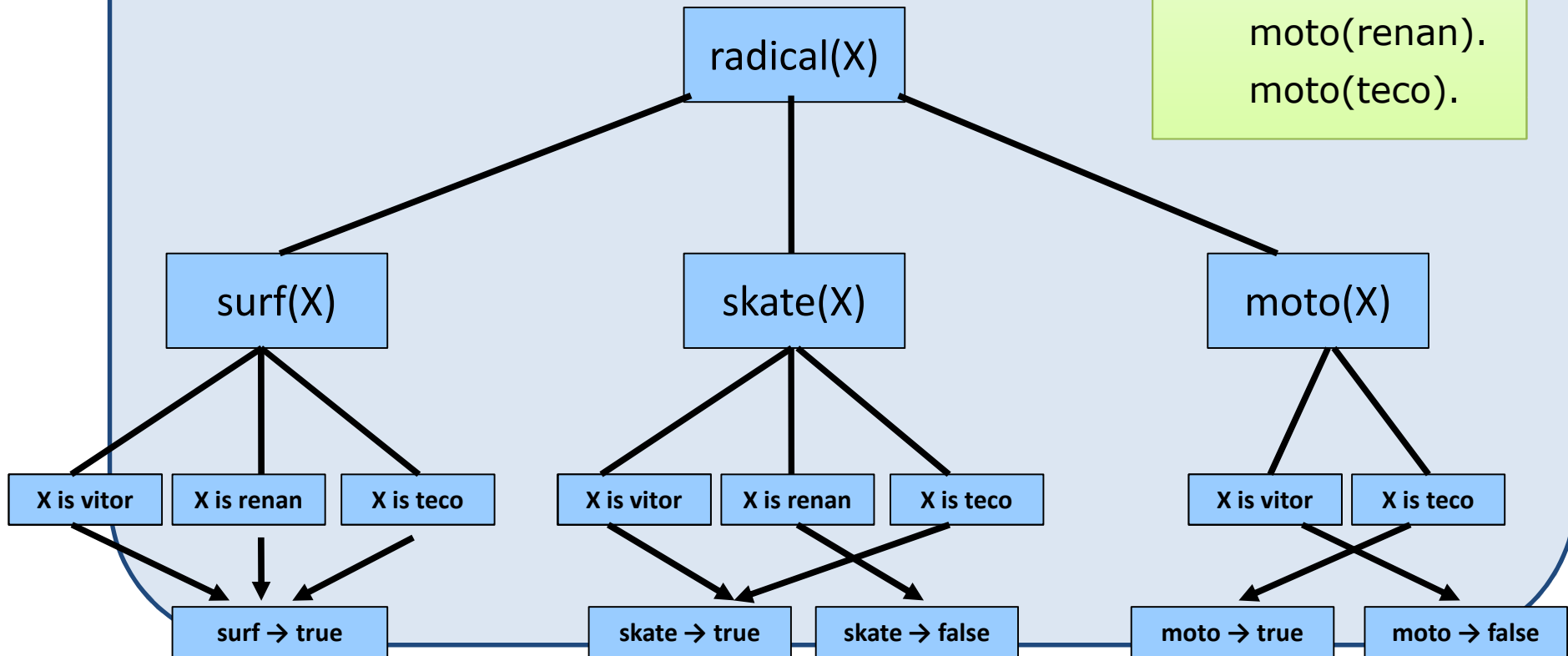


Fluxo de controle implícito

❑ Fluxo do percurso:

`radical(X):- surf(X), skate(X), moto(X).`

`surf(vitor).`
`surf(renan).`
`surf(teco).`
`skate(vitor).`
`skate(teco).`
`moto(renan).`
`moto(teco).`



Exercício

- Implemente os exercícios vistos em aula
- Implemente os exercícios presentes na apostila