

Laboratório de PROLOG #1/3

Disciplina de Modelos de Linguagens de Programação

Modelo lógico

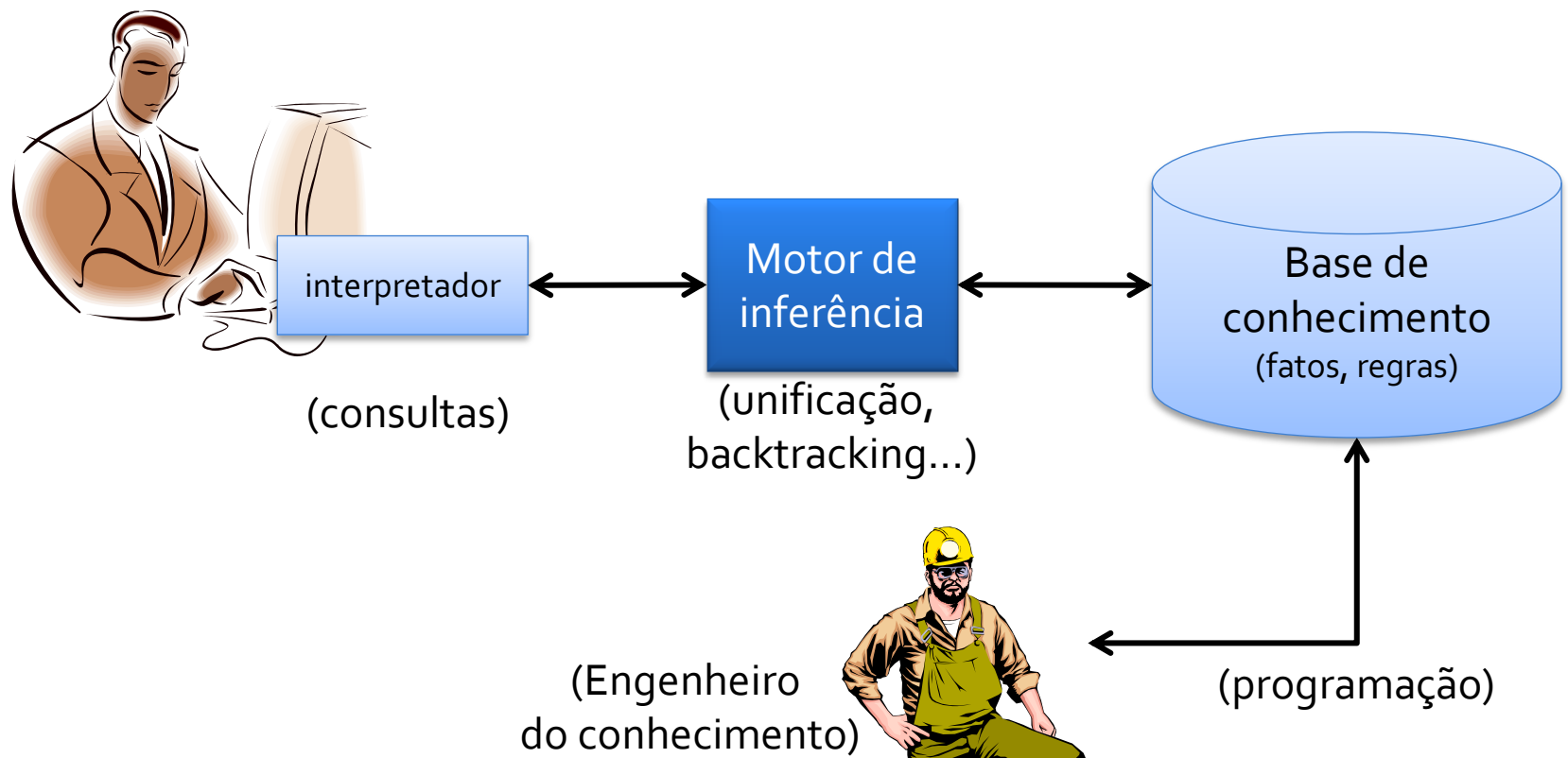
Tópicos da aula

- Programação em lógica
- O ambiente
- Fundamentos PROLOG
 - Fatos
 - Consultas
 - Regras
 - Átomos versus cláusulas versus variáveis

Programação em lógica

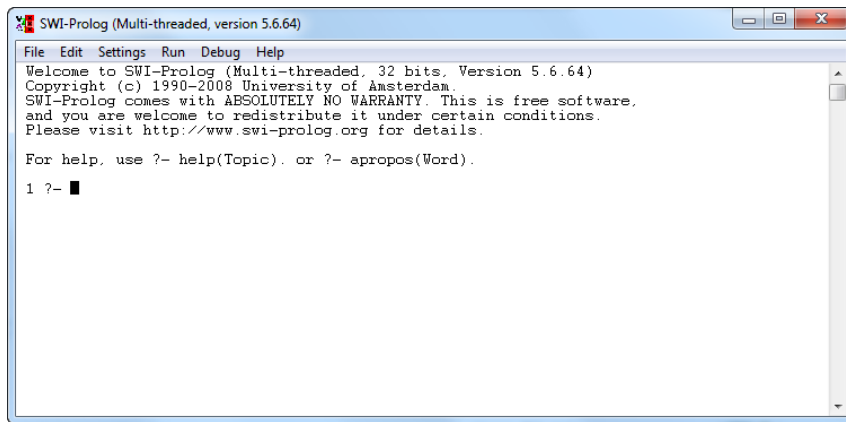
- **PROLOG = Programmation en Logique**
 - Desenvolvida por Philippe Rouseel e Colmerauer
 - Existem diferentes dialetos e implementações, que diferem na sintaxe e nas bibliotecas (Arity Prolog, Visual Prolog, Turbo Prolog, Strawberry Prolog, LPA Prolog, [SWI Prolog](#)...)

Arquitetura



Ambiente utilizado

SWI PROLOG



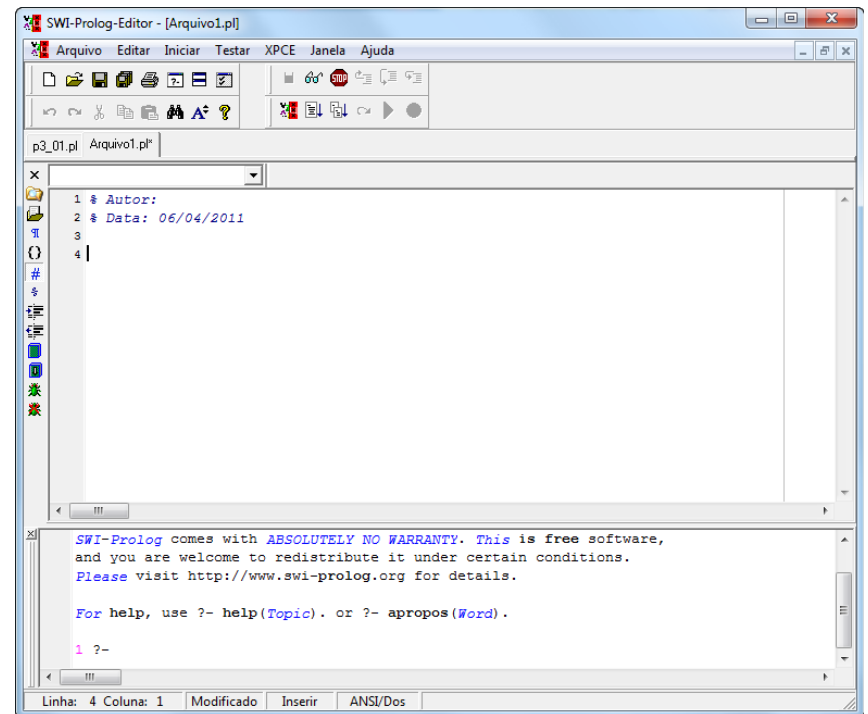
The screenshot shows the SWI-Prolog command-line interface. The window title is "SWI-Prolog (Multi-threaded, version 5.6.64)". The menu bar includes "File", "Edit", "Settings", "Run", "Debug", and "Help". The main text area displays a welcome message: "Welcome to SWI-Prolog (Multi-threaded, 32 bits, 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." Below this, it says "For help, use ?- help(Topic). or ?- apropos(Word)." The prompt "1 ?- " is followed by a cursor.

```
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 ?- █
```

SWI-PROLOG EDITOR



The screenshot shows the SWI-Prolog Editor graphical interface. The window title is "SWI-Prolog-Editor - [Arquivo1.pl]". The menu bar includes "Arquivo", "Editar", "Iniciar", "Testar", "XPCE", "Janela", and "Ajuda". The toolbar contains various icons for file operations and editing. The main text area shows a Prolog file with the following content: "1 # Autor:", "2 # Data: 06/04/2011", "3", "4 |". The status bar at the bottom indicates "Linha: 4 Coluna: 1", "Modificado", "Inserir", and "ANSI/Dos".

```
SWI-Prolog-Editor - [Arquivo1.pl]
Arquivo Editar Iniciar Testar XPCE Janela Ajuda
p3_01.pl Arquivo1.pl*
1 # Autor:
2 # Data: 06/04/2011
3
4 |
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).

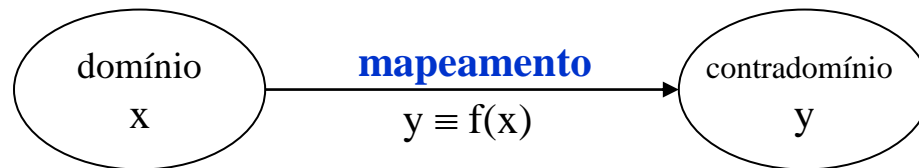
1 ?-
Linha: 4 Coluna: 1 Modificado Inserir ANSI/Dos
```

<http://www.swi-prolog.org/>

Fundamentos

■ Modelos Imperativo e Funcional

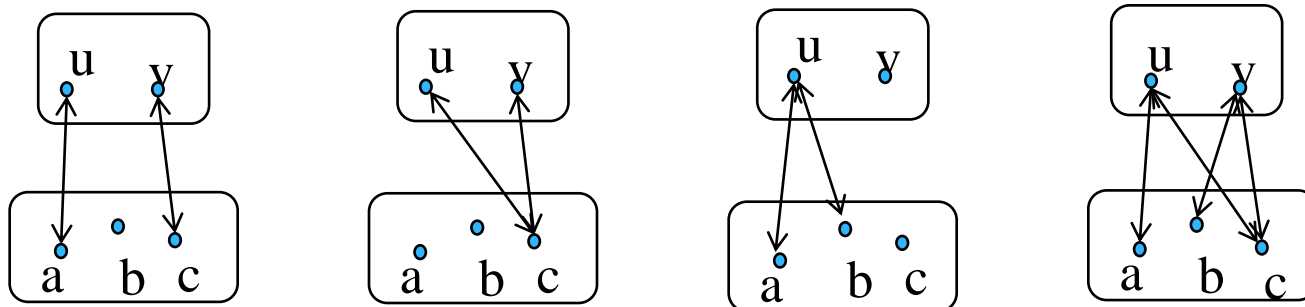
Implementam um **mapeamento** de entradas em saídas:



- Uma entrada gera uma saída (mapeamento 1 para 1)

■ Modelo Lógico

Implementa uma **relação** (que pode ser de vários tipos):

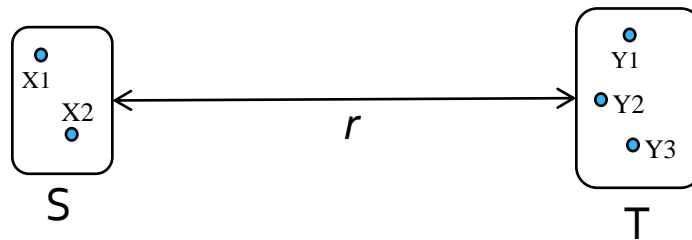


- Um elemento não gera outro, mas, sim, se relaciona com outro. A relação é mais geral do que o mapeamento, sendo o modelo potencialmente mais abstrato do que os outros

Fundamentos: relações

■ Relação:

Sejam dois conjuntos de valores S e T :



Ou seja:

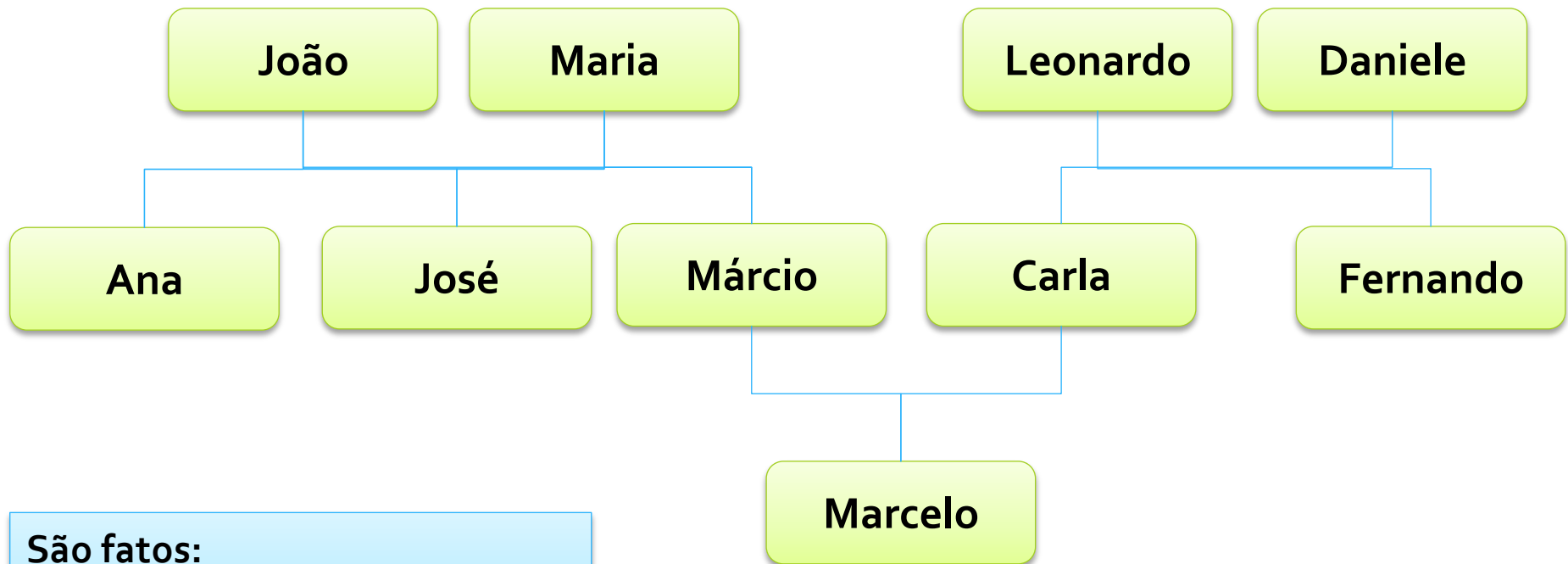
- uma relação é estabelecida entre objetos e resulta em verdadeiro ou falso

Observações:

- não há distinção entre entradas e saídas
- y não é gerado a partir de x (uma entrada não é transformada em saída)

Fundamentos: fatos

Considere a seguinte árvore genealógica:



São fatos:

- Pessoas
- Sexo das pessoas
- Parentesco entre pais e filhos

Fatos são especificados através de cláusulas Horn incondicionais

Fundamentos: cláusula Horn

- Homenagem a Alfred Horn, que estudou o significado de tais cláusulas em 1951
- Clausula Horn:

$$A_0 \leftarrow A_1 \wedge A_2 \dots \wedge A_n$$

Cláusula Horn
(um único literal positivo)

- Lê-se: A_0 (é verdade) *se* A_1 *e* A_2 ... *e* A_n
- Exemplo: `natural(N) :- inteiro(N), N>0.`

Fundamentos: fato

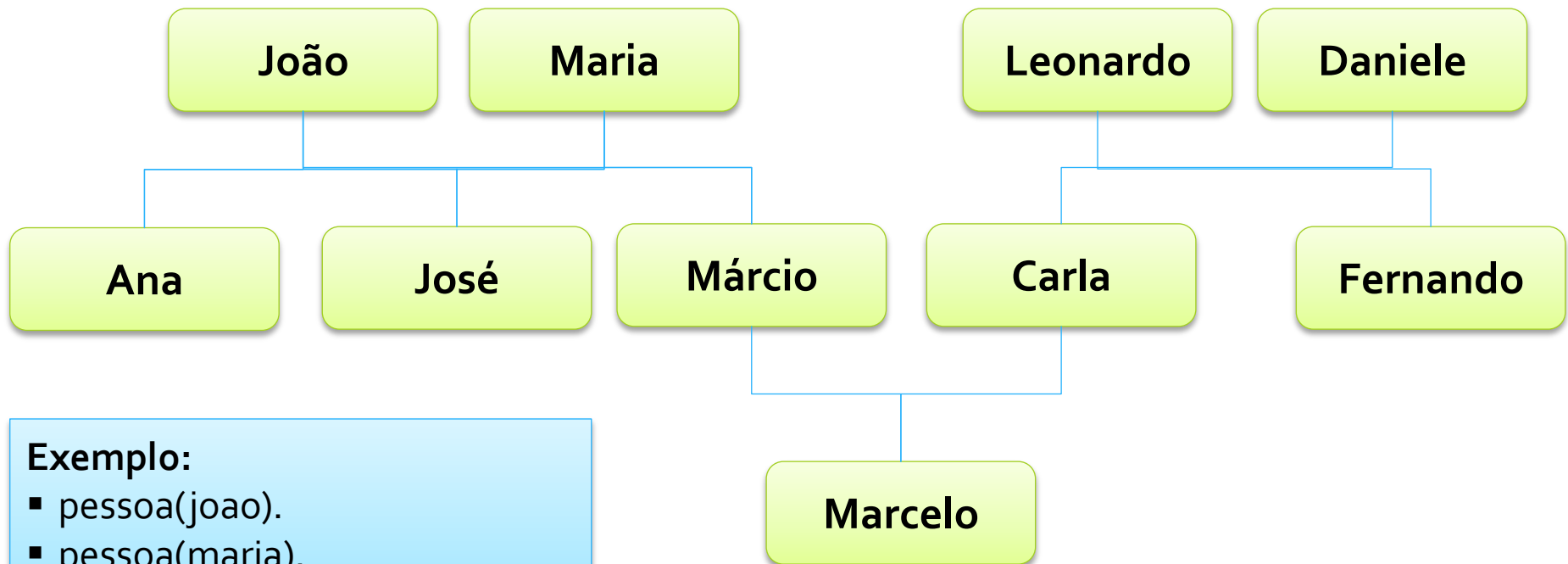
- Clausula Horn sem condições (incondicional):

$$A_0 \leftarrow A_1 \wedge A_2 \wedge \dots \wedge A_n$$

- Exemplo: `inteiro(10)`.
- Observações:
 - Prolog não tem tipos de dados como as outras linguagens
 - Todos os dados são do mesmo tipo: o termo
 - Termos representam qualquer coisa e sua natureza depende de como eles foram declarados
 - Termos são todos em minúsculas e sem espaço!

Exercício

- Especifique fatos que representem a árvore seguinte, considerando pessoas e seus respectivos sexos:



Exemplo:

- `pessoa(joao).`
- `pessoa(maria).`
- `masculino(joao).`
- `feminino(maria).`

Fundamentos: consultas

- Consultas simples:

- `pessoa(leandro) .` % Leandro é uma pessoa?

- Conhecimento sobre o mundo :

- `pessoa(X) .` % Quais pessoas existem?

- Atenção:

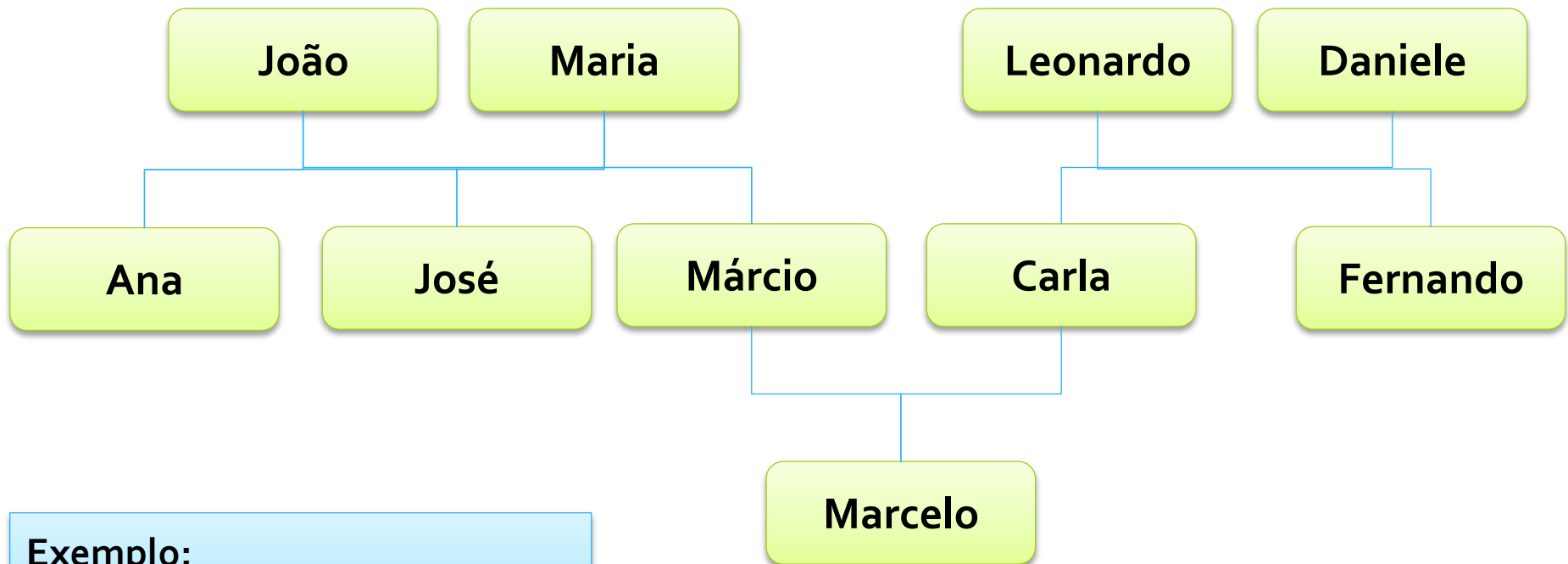
- qualquer termo com letra maiúscula é uma variável
- uma consulta pode ter uma única resposta (sim/não), várias ou nenhuma

Fundamentos: consultas

- Cuidado com a **Hipótese do Mundo Fechado**:
 - programa é considerado um “mundo fechado”
(tudo que a máquina sabe deve ser definida nele)
 - o que não se sabe ser verdadeiro é considerado falso; o que não se pode provar é considerado falso (o que não significa que isso seja verdade!)
 - cuidado com as negações!

Exercício

- Considerando a mesma árvore, especifique fatos que representem relações entre pais e filhos :



Exemplo:

- pai(ana, joao).
- mae(ana, maria).

Exercício

- Elabore consultas para:
 - Determinar se relações são válidas:
 - João é pai de José?
 - Encontrar elementos que satisfaçam relações:
 - Quem é o pai de José?
 - Saber se há um elemento que satisfaça uma relação:
 - Há pais? Filhos?

Fundamentos: consultas

- PROLOG permite especificar e consultar:
 - Dados a e b , determinar se $R(a,b)$ é verdadeira
 - Dado a , encontrar todo y tal que $R(a,y)$ é verdadeira
 - Dado b , encontrar todo x tal que $R(x,b)$ é verdadeira
 - Encontrar todo x e todo y tal que $R(x,y)$ é verdadeira

Fundamentos: regras

- Descrevem as relações através de condições e conectivos lógicos, descritas por cláusulas Horn
- Exemplo:

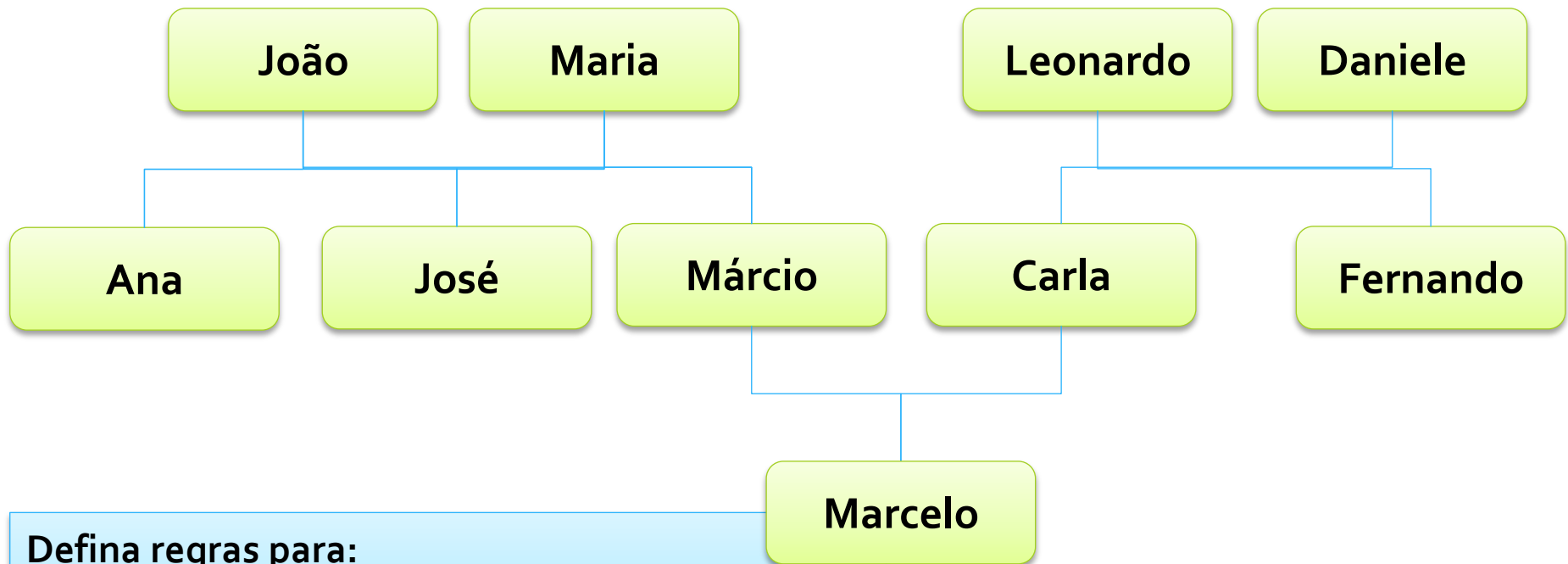
```
pai(P, X) :- genitor(P,X) , masculino(P).  
mae(M, X) :- genitor(M,X) , feminino(M).
```

Fundamentos: operadores

Operador	Significado
,	And
;	Or
=	Unificação
\=	Negação da unificação
==	Teste de identidade
\==	Negação da identidade
:=	Igualdade aritmética
< > >= =<	Operadores relacionais
:=	Condicional

Fundamentos: regras

Levando em conta a árvore anterior:



Defina regras para:

- Inferir tios (independente do sexo)
- Inferir avós (independente do sexo)
- Inferir irmãos (independente do sexo)

Fundamentos: termos

■ Termos simples:

- Átomos:
 - Constantes alfanuméricas: leonardo, 'Porto Alegre'
 - Constantes numéricas: 1, 12.12
- Variáveis: X, Cidades, _ruas, _123abc

■ Termos funcionais:

- Fatos/consultas: pessoa(pedro, 22, masculino).
- Regras: capital_pais(X,Y) :- pais(X), cidade(Y), capital(X,Y).

Observações sobre PROLOG:

- **não é tipada**, só havendo termos simples (átomos, variáveis) ou funcionais (fatos, regras)
- **diferencia maiúsculas** (variáveis) **de minúsculas** (átomos)

Fundamentos: programação

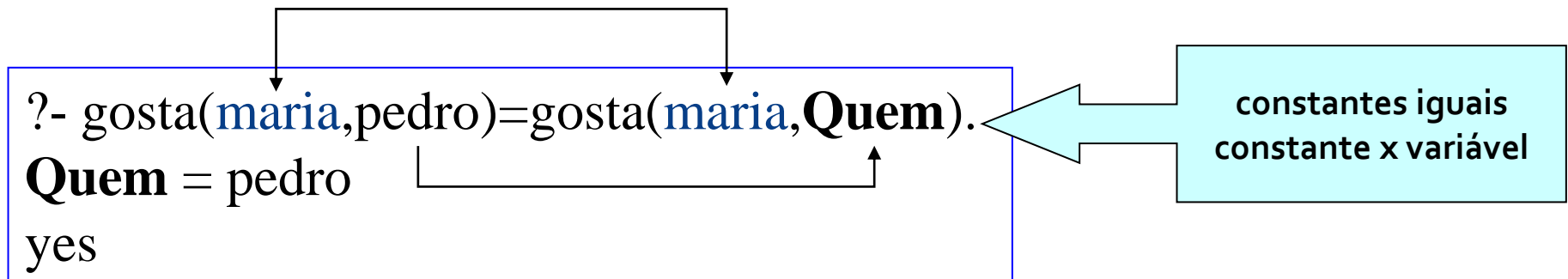
- **Consiste em especificar uma base de conhecimento (fatos e regras):**
 - `capital('Brasil', 'Brasilia').`
 - `capital('Franca', 'Paris').`
 - `capital('RS', 'Porto Alegre').`
 - `capital('SC', 'Florianopolis').`
 - `estadode('Brasil', 'SC').`
 - `estadode('Brasil', 'RS').`
 - `estadode('Franca', 'Bretanha').`
 - `estadode('Franca', 'Normandia').`
 - `capital_pais(X, Y) :- pais(X), cidade(Y), capital(X,Y).`
 - `capital_estado(X,Y):-estado(X), cidade(Y), capital(X,Y).`

Processo de inferência em Prolog

- Para provar um objetivo, o processo de inferência deve encontrar uma cadeia de regras e/ou fatos na base de conhecimento que conecte o objetivo a um ou mais fatos
- Se Q é o objetivo, o sistema procura por Q como um fato na base:
 - Se encontra → Sucesso (verdadeiro)
 - Se não encontra: procura por uma sequência de proposições que leve a Q
- **Mecanismos:** *unificação*, resolução *top-down* e busca em profundidade (*depth-first*) com retrocesso (*backtracking*)

Consultas e unificação

- Dados dois termos, diz-se que eles se unificam caso:
 - ambos sejam constantes e iguais;
 - um dos termos seja uma variável (instanciação);
 - ambos sejam cláusulas de mesmo predicado (mesmo nome e aridade) e os argumentos se unificam



Teste o exemplo diretamente no prompt!

Exercício

Data a Base de Dados: **Elabore as seguintes consultas:**

gosta(pedro, leitura).
gosta(maria, leitura).
gosta(paulo, leitura).
gosta(pedro, cinema).
gosta(paulo, cinema).
gosta(vera, cinema).
gosta(paulo, boliche).
gosta(maria, boliche).
gosta(vera, boliche).
gosta(pedro, maria).
gosta(maria, peixe).
gosta(maria, vinho).
gosta(pedro, vinho).

- a) Paulo gosta de ler?**
- b) Quem gosta de boliche?**
- c) Quem gosta de boliche e também de cinema?**
- d) pedro e maria se gostam?**
- e) Quais são as coisas que Pedro e Maria (ambos) gostam?**
- f) Existe algo que paulo goste? Sim ou não?**

Leitura recomendada

- Definição de Prolog na Wikipedia
<http://pt.wikipedia.org/wiki/Prolog>
- PROLOG tutorials by James Power
<http://www.cs.nuim.ie/~jpower/Courses/PROLOG/>
- Capítulo de linguagens de programação lógicas do livro: Sebesta. Robert W. Conceitos de Linguagens de Programação. Bookman: Porto Alegre, 2000.