

# INTERPRETADOR DE FÓRMULAS DO CÁLCULO PROPOSICIONAL

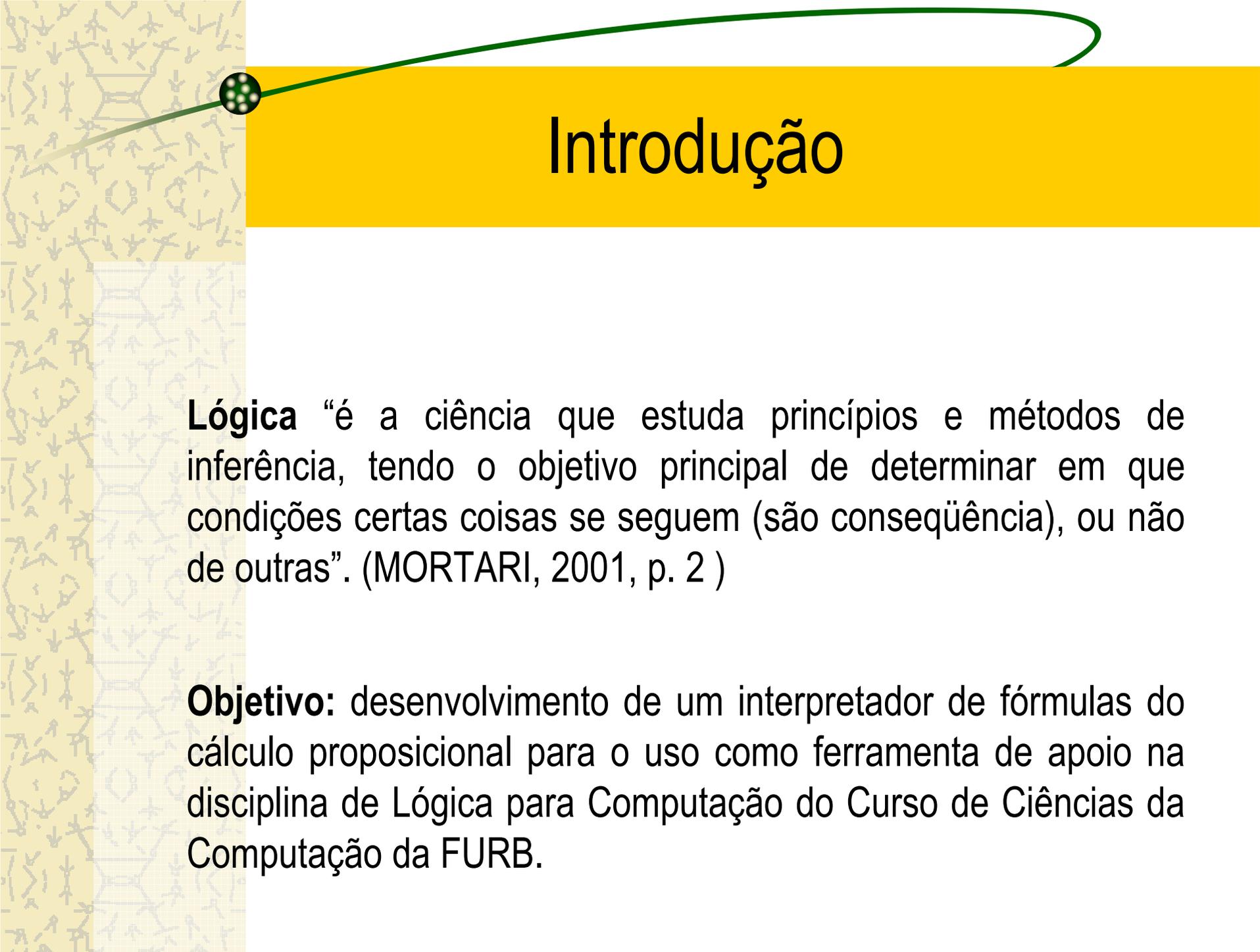
Acadêmica: Michele Milane Tambosi

Orientadora: Joyce Martins



# Roteiro

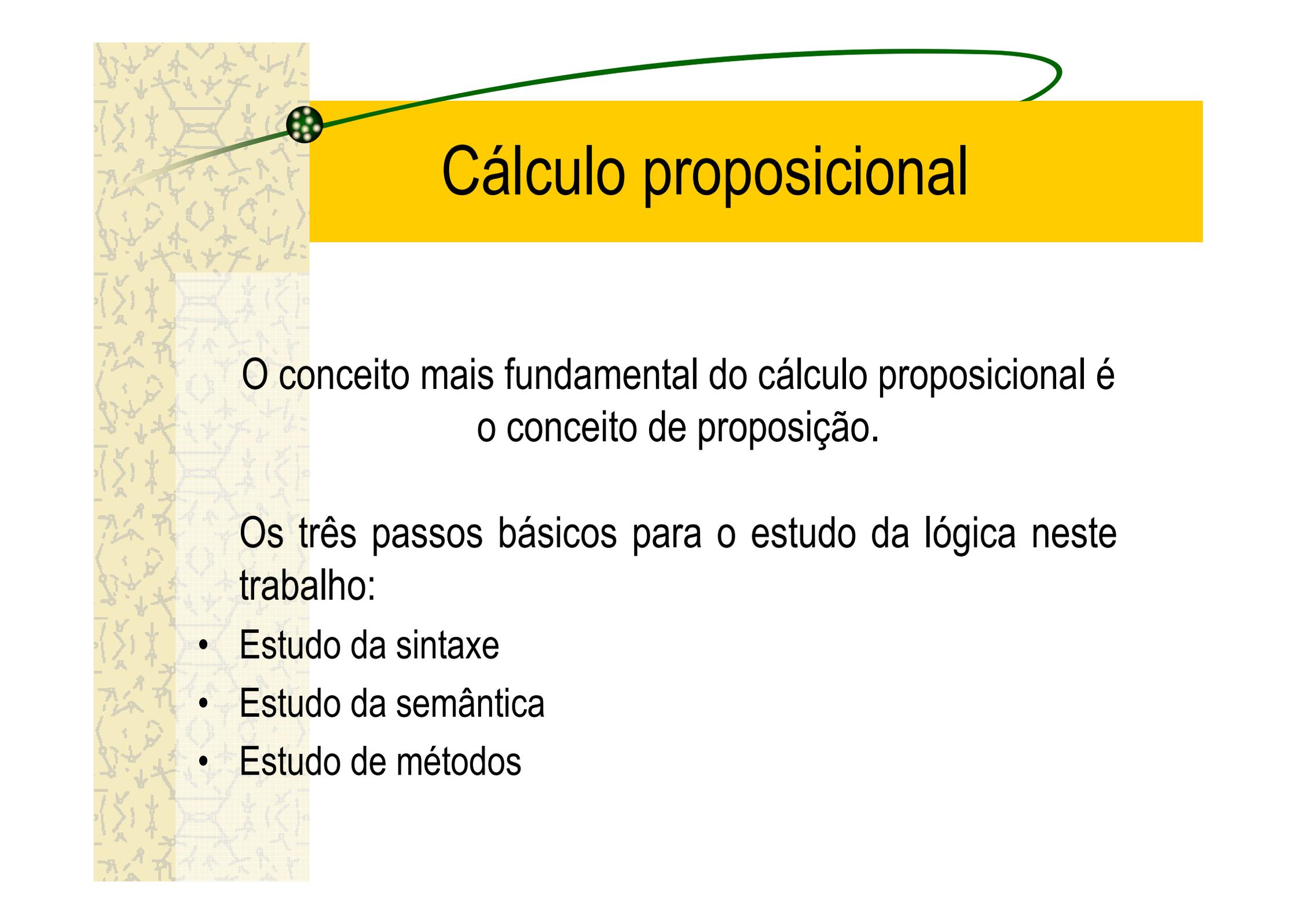
- ▶ **Introdução**
- ▶ **Lógica Proposicional**
  - Sintaxe**
  - Semântica**
  - Métodos**
- ▶ **Processadores de linguagens**
  - Estrutura de um tradutor
    - Análise léxica**
    - Análise sintática**
    - Análise semântica**
    - Geração de código intermediário**
- ▶ **Desenvolvimento**
  - Especificação
  - Implementação
- ▶ **Conclusões**
  - Extensões



# Introdução

**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”. (MORTARI, 2001, p. 2 )

**Objetivo:** desenvolvimento de um interpretador de fórmulas do cálculo proposicional para o uso como ferramenta de apoio na disciplina de Lógica para Computação do Curso de Ciências da Computação da FURB.



# Cálculo proposicional

O conceito mais fundamental do cálculo proposicional é o conceito de proposição.

Os três passos básicos para o estudo da lógica neste trabalho:

- Estudo da sintaxe
- Estudo da semântica
- Estudo de métodos

# Cálculo proposicional: sintaxe

## Alfabeto:

- ☛ símbolos de pontuação: ( )
- ☛ símbolos de verdade: *true* e *false*
- ☛ símbolos proposicionais:  $P, Q, R, S, P_1, Q_1, R_1, S_1, P_2, Q_2, R_2, S_2, \dots$
- ☛ conectivos proposicionais:  $\neg$  (negação),  $\wedge$  (conjunção),  $\vee$  (disjunção),  $\rightarrow$  (implicação) e  $\leftrightarrow$  (equivalência)

# Cálculo proposicional: sintaxe

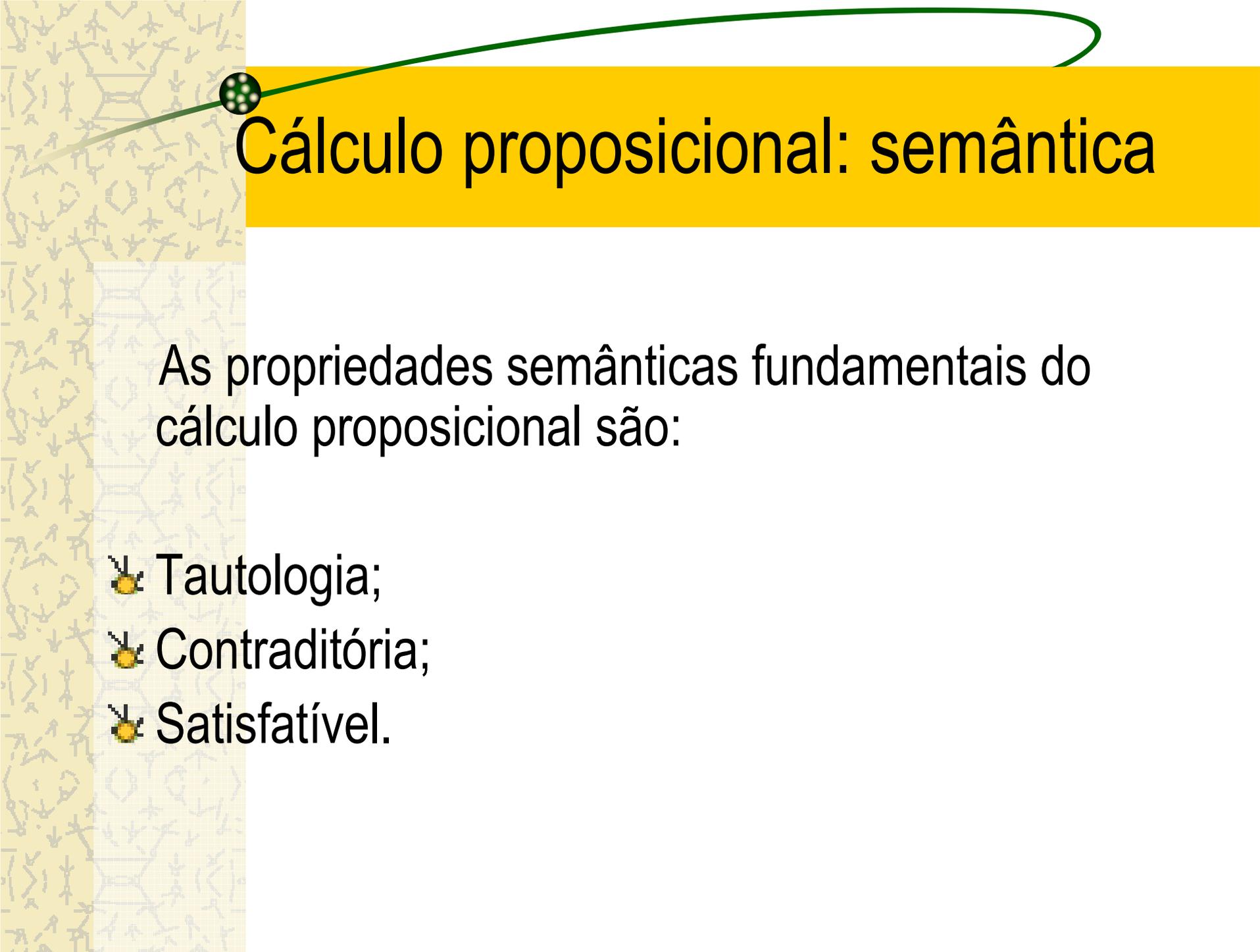
## Fórmula bem-formada (fbf) (SOUZA, 2002):

- ✚ todo símbolo de verdade é uma fórmula;
- ✚ todo símbolo proposicional é uma fórmula;
- ✚ se H e G são fórmulas, então:
  - (  $\neg$  H ) é uma fórmula: negação
  - ( H  $\wedge$  G ) é uma fórmula: conjunção
  - ( H  $\vee$  G ) é uma fórmula: disjunção
  - ( H  $\rightarrow$  G ) é uma fórmula: H é o antecedente e G é o conseqüente
  - ( H  $\leftrightarrow$  G ) é uma fórmula: H é o lado esquerdo e G é o lado direito

# Cálculo proposicional: semântica

A semântica do cálculo proposicional associa a cada objeto sintático um significado.

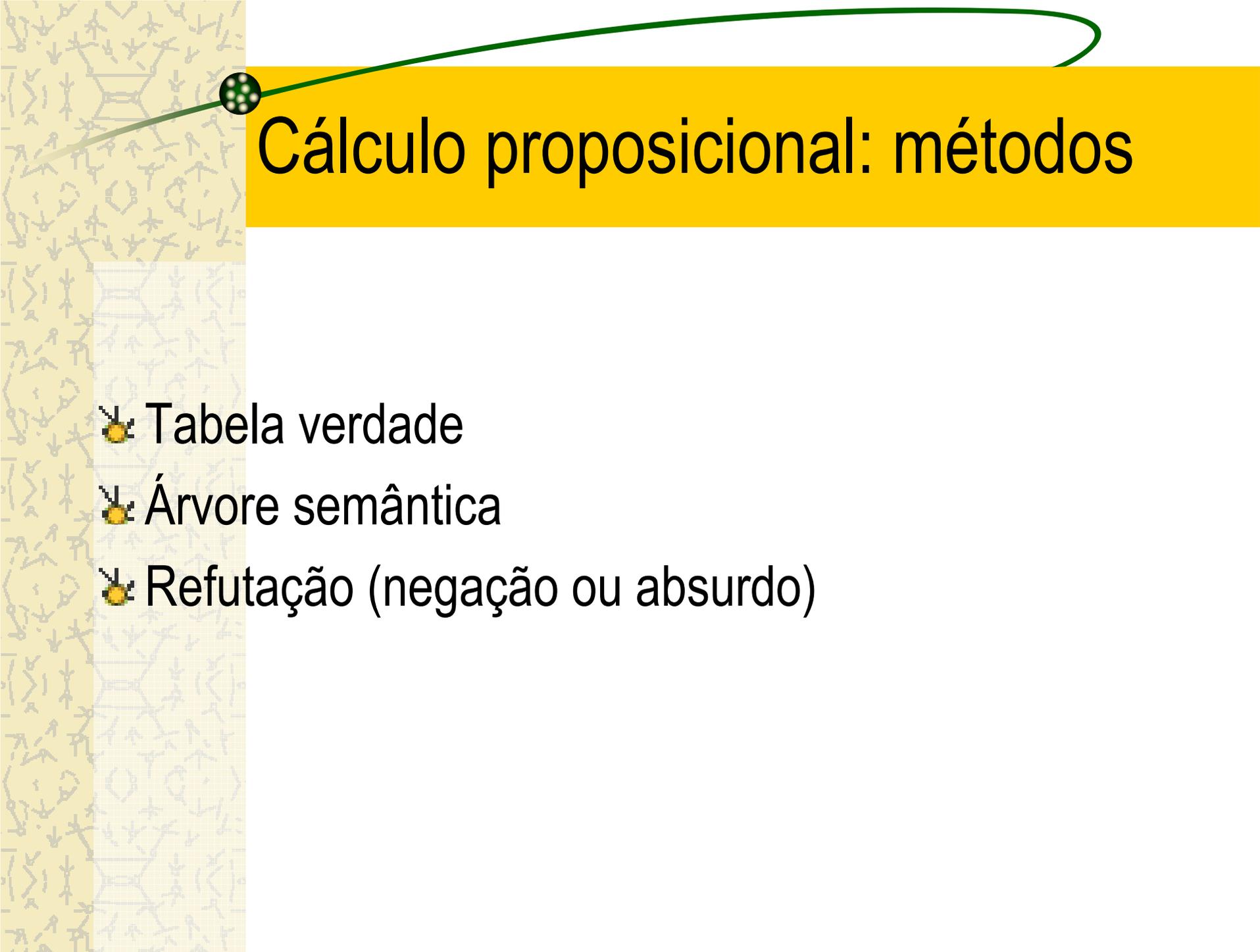
P	Q	$\neg P$	$P \vee Q$	$P \wedge Q$	$P \rightarrow Q$	$P \leftrightarrow Q$
T	T	F	T	T	T	T
T	F	F	T	F	F	F
F	T	T	T	F	T	F
F	F	T	F	F	T	T



# Cálculo proposicional: semântica

As propriedades semânticas fundamentais do cálculo proposicional são:

- ✖ Tautologia;
- ✖ Contraditória;
- ✖ Satisfatível.



# Cálculo proposicional: métodos

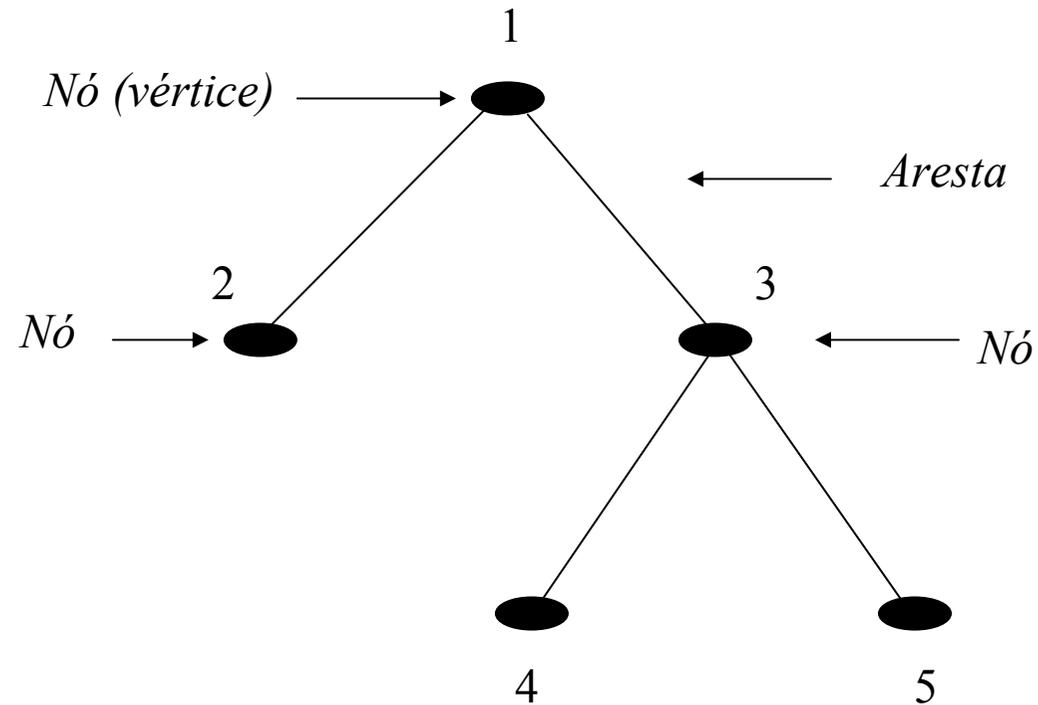
- ✦ Tabela verdade
- ✦ Árvore semântica
- ✦ Refutação (negação ou absurdo)

# Cálculo proposicional: tabela verdade

$$((P \rightarrow Q) \rightarrow (\neg P \vee Q))$$

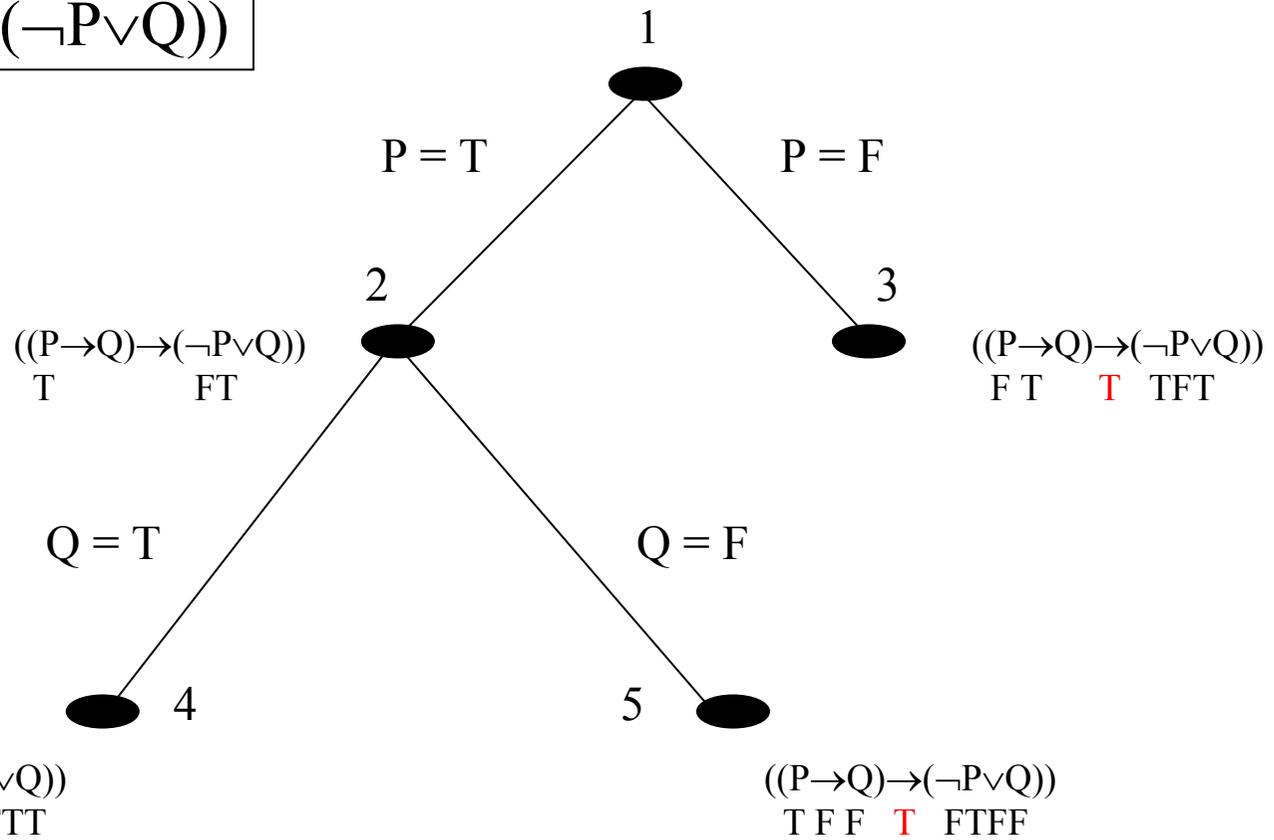
P	Q	$\neg P$	$P \rightarrow Q$	$\neg P \vee Q$	$((P \rightarrow Q) \rightarrow (\neg P \vee Q))$
T	T	F	T	T	T
T	F	F	F	F	T
F	T	T	T	T	T
F	F	T	T	T	T

# ● Cálculo proposicional: árvore semântica



# ● Cálculo proposicional: árvore semântica

$$((P \rightarrow Q) \rightarrow (\neg P \vee Q))$$

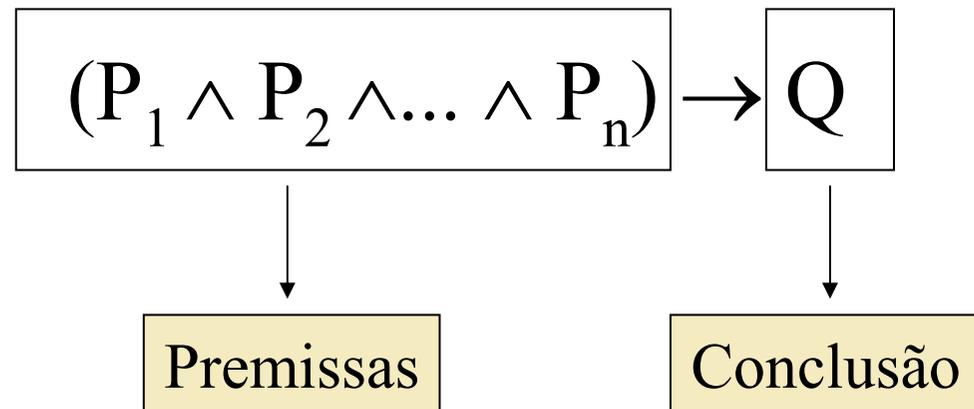


# Cálculo proposicional: refutação

$$((P \rightarrow Q) \rightarrow (\neg P \vee Q))$$

$((P$	$\rightarrow$	$Q)$	$\rightarrow$	$(\neg$	$P$	$\vee$	$Q))$
T	T	F	F	F	T	F	F
5	2	5	1	3	4	2	3

# Cálculo proposicional: argumentos



# Cálculo proposicional: argumento

*Se meu cliente fosse culpado, a faca estaria na gaveta. Ou a faca não estava na gaveta ou Jacson Pritchard viu a faca. Se a faca não estava lá no dia 10 de outubro, então Jacson Pritchard não viu a faca. Além disso, se a faca estava lá no dia 10 de outubro, então a faca estava na gaveta e o martelo estava no celeiro. Mas todos sabemos que o martelo não estava no celeiro. Portanto, senhoras e senhores, meu cliente é inocente. (GERSTING, 2001, p. 1)*

Pergunta-se: O cliente é inocente?

- ✚  $P_1$ : significa “meu cliente é culpado”;
- ✚  $P_2$ : significa “a faca estaria na gaveta”;
- ✚  $P_3$ : significa “Jacson Pritchard viu a faca”;
- ✚  $P_4$ : significa “a faca estava lá no dia 10 de outubro” ;
- ✚  $P_5$ : significa “o martelo estava no celeiro”;

$$((P_1 \rightarrow P_2) \wedge (\neg(\neg P_2 \leftrightarrow P_3))) \wedge (\neg P_4 \rightarrow \neg P_3) \wedge (P_4 \rightarrow (P_2 \wedge P_5)) \wedge \neg P_5 \rightarrow \neg P_1$$

# Processadores de linguagens

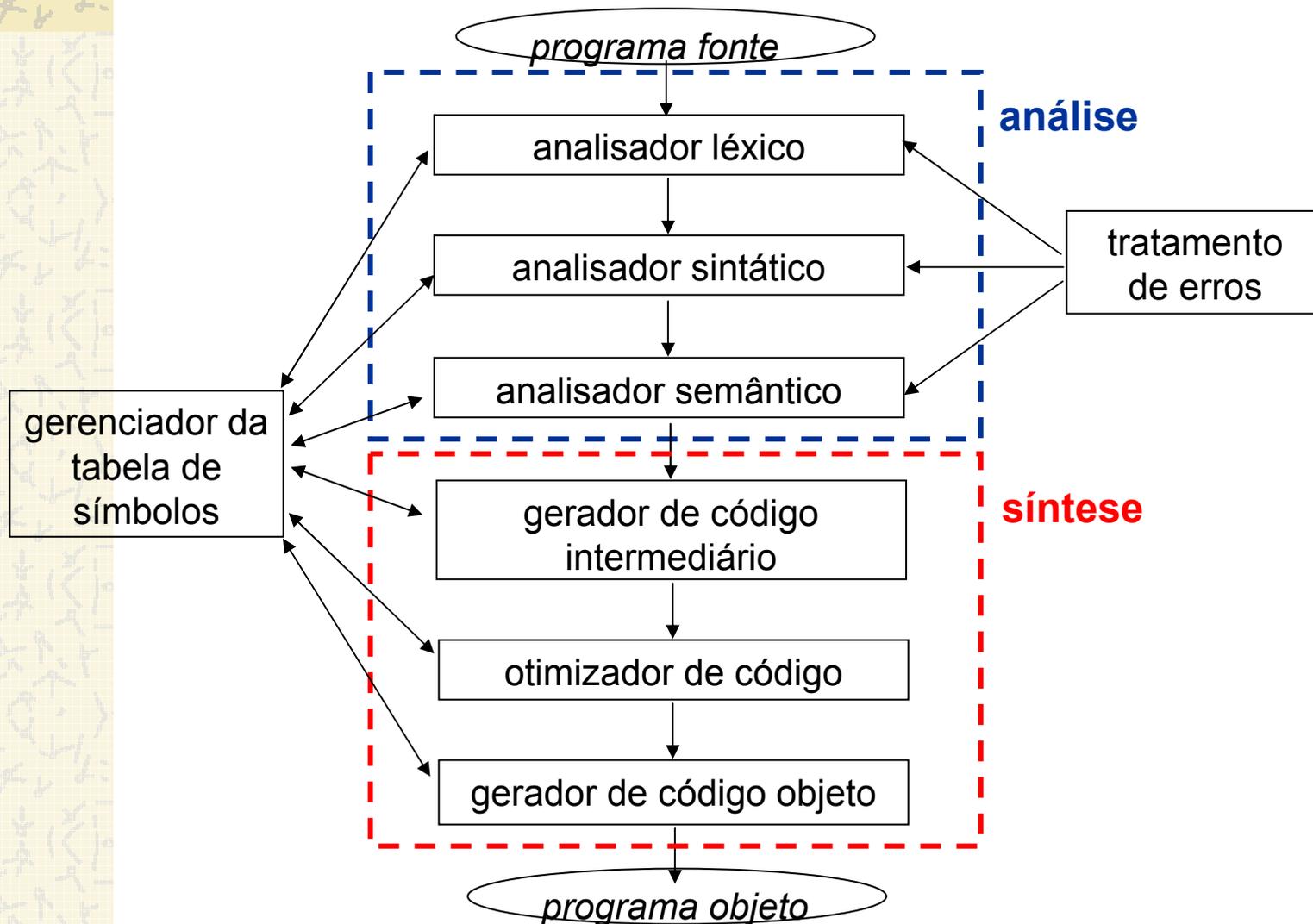
Linguagem fonte

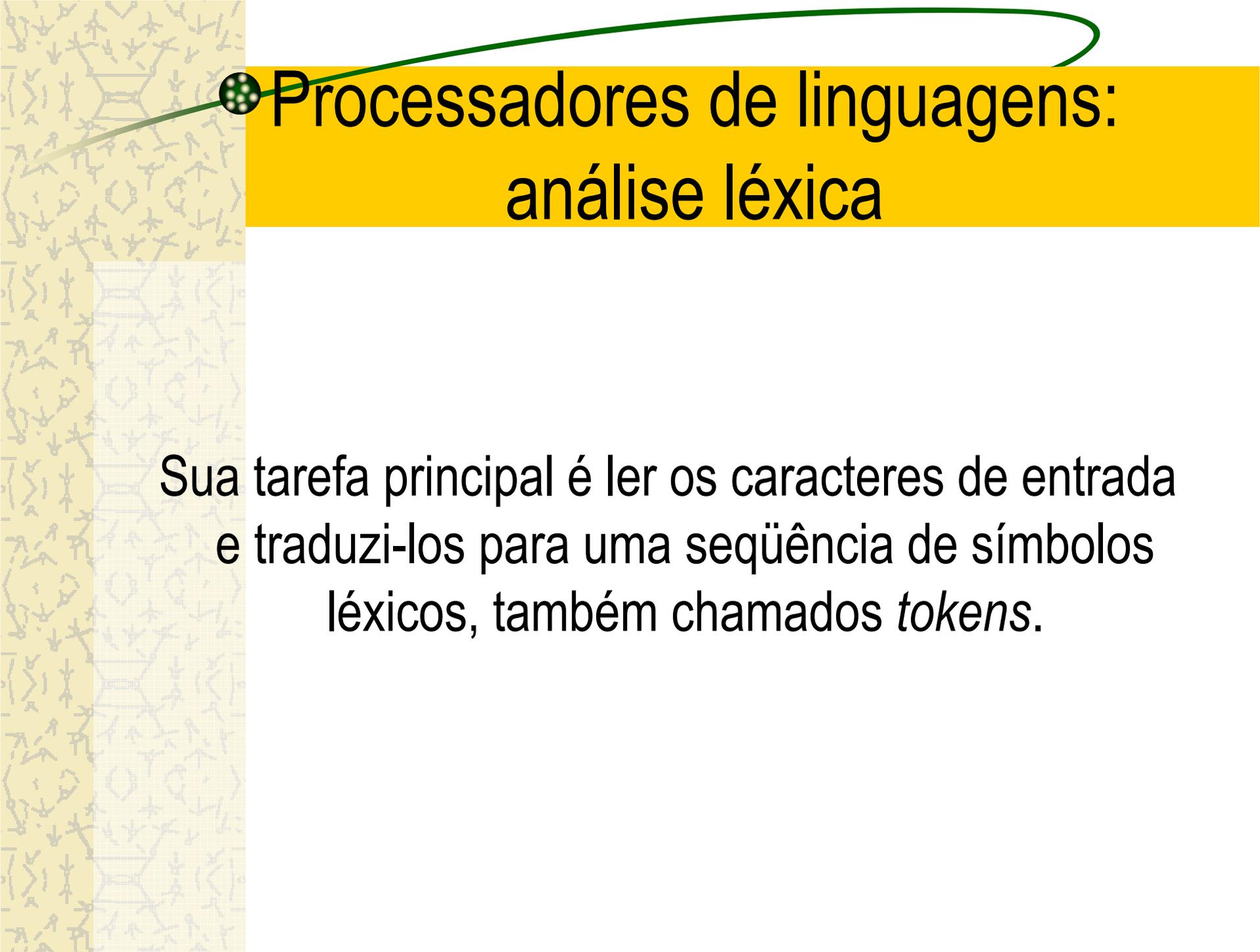
*TRADUTOR*

Linguagem objeto



# Processadores de linguagens





## ● Processadores de linguagens: análise léxica

Sua tarefa principal é ler os caracteres de entrada e traduzi-los para uma seqüência de símbolos léxicos, também chamados *tokens*.

# ● Processadores de linguagens: especificação dos tokens

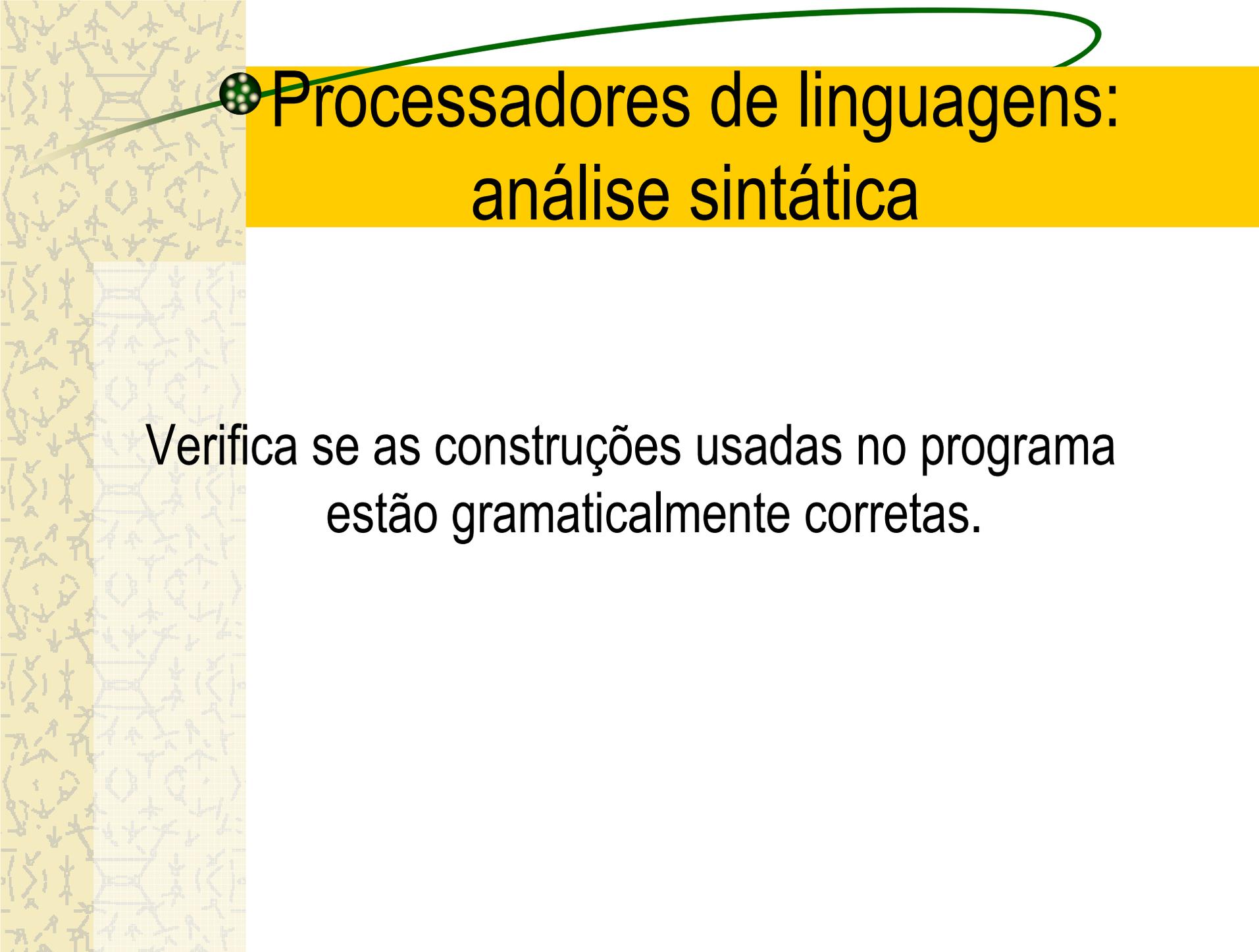
## Expressão regular

```
<símbolo proposicional> ::= <letra> <dígito>*
```

## Definições regulares

```
<dígito> ::= 0 | 1 | ... | 9
```

```
<letra> ::= P | Q | R | S
```



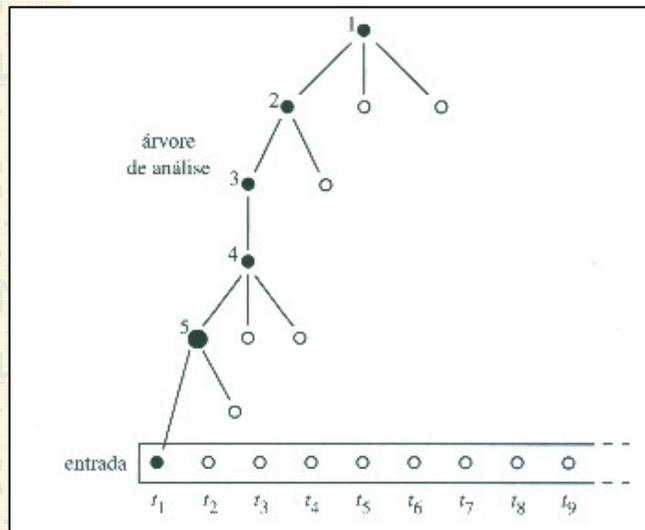
## ● Processadores de linguagens: análise sintática

Verifica se as construções usadas no programa estão gramaticalmente corretas.

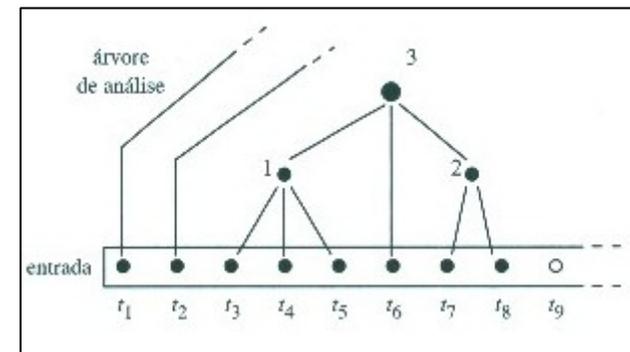
# ● Processadores de linguagens: especificação das regras sintáticas

```
<fórmula> ::= <símbolo proposicional>  
            | <símbolo verdade>  
            | (<fórmula> <conectivo> <fórmula>)  
            | ¬ <fórmula>  
  
<conectivo> ::= ∨ | ∧ | → | ↔  
  
<símbolo verdade> ::= T | F
```

# ● Processadores de linguagens: tipos de análise sintática



*Top-down* (descendente)



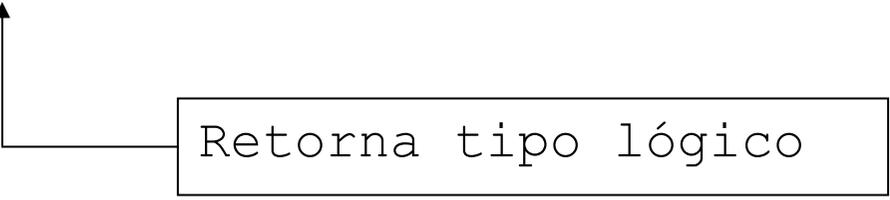
*Bottom-up* (reduitiva)

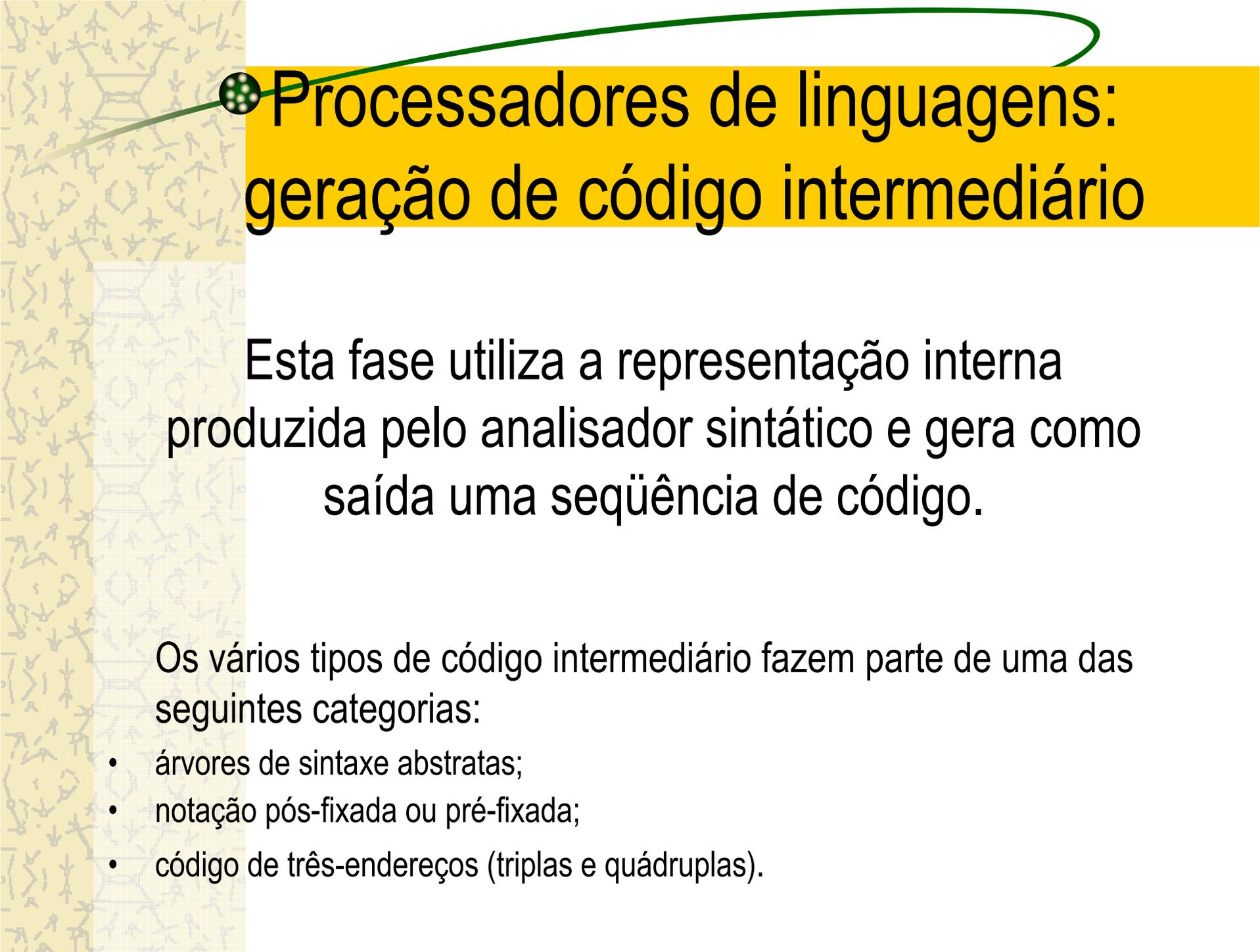
# ● Processadores de linguagens: análise semântica

Sua principal função é verificar se o programa não possui erros de significado.

```
while <expressão> do <comando> ;
```

Retorna tipo lógico





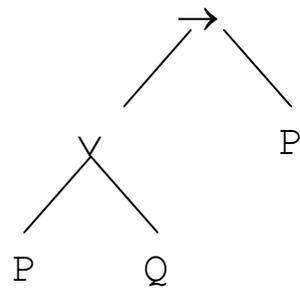
## ● Processadores de linguagens: geração de código intermediário

Esta fase utiliza a representação interna produzida pelo analisador sintático e gera como saída uma seqüência de código.

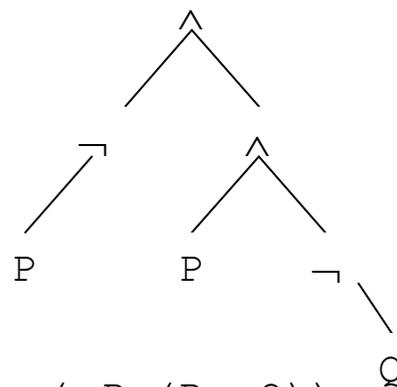
Os vários tipos de código intermediário fazem parte de uma das seguintes categorias:

- árvores de sintaxe abstratas;
- notação pós-fixada ou pré-fixada;
- código de três-endereços (triplas e quádruplas).

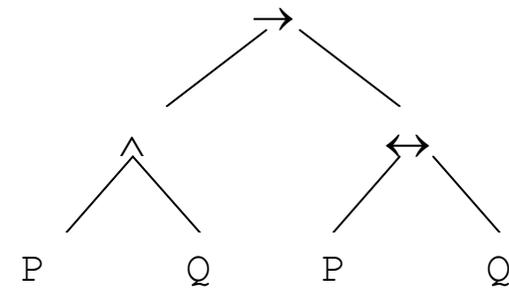
# Processadores de linguagens: árvores de sintaxe



$((P \vee Q) \rightarrow P)$



$(\neg P \wedge (P \wedge \neg Q))$



$((P \wedge Q) \rightarrow (P \leftrightarrow Q))$

# Processadores de linguagens: Notação pós-fixada ou pré-fixada

Notação		
<i>Infixada</i>	<i>pós-fixada</i>	<i>pré-fixada</i>
$((P \vee Q) \rightarrow P)$	$PQ \vee P \rightarrow$	$\rightarrow \vee PQP$
$(\neg P \wedge (P \wedge \neg Q))$	$P \neg PQ \neg \wedge \wedge$	$\wedge \neg P \wedge P \neg Q$
$((P \wedge Q) \rightarrow (P \leftrightarrow Q))$	$PQ \wedge PQ \leftrightarrow \rightarrow$	$\rightarrow \wedge PQ \leftrightarrow PQ$



# Especificação da Linguagem do Cálculo Proposicional

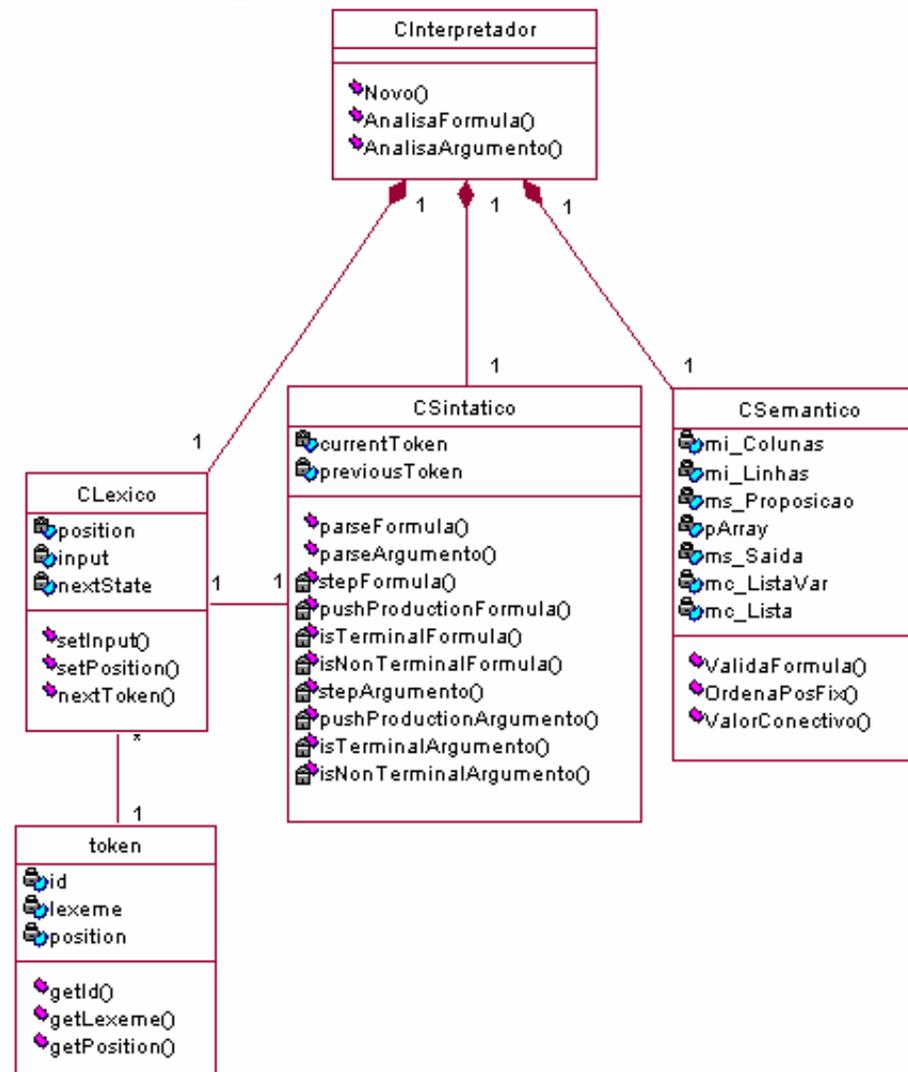
## Definições regulares de fórmulas do cálculo proposicional

```
<dígito> ::= 0 | 1 | ... | 9  
<letra> ::= P | Q | R | S  
<símbolo proposicional> ::= <letra> <dígito>*  
<conectivo> ::= ~ | | | & | -> | <->  
<símbolo verdade> ::= T | F
```

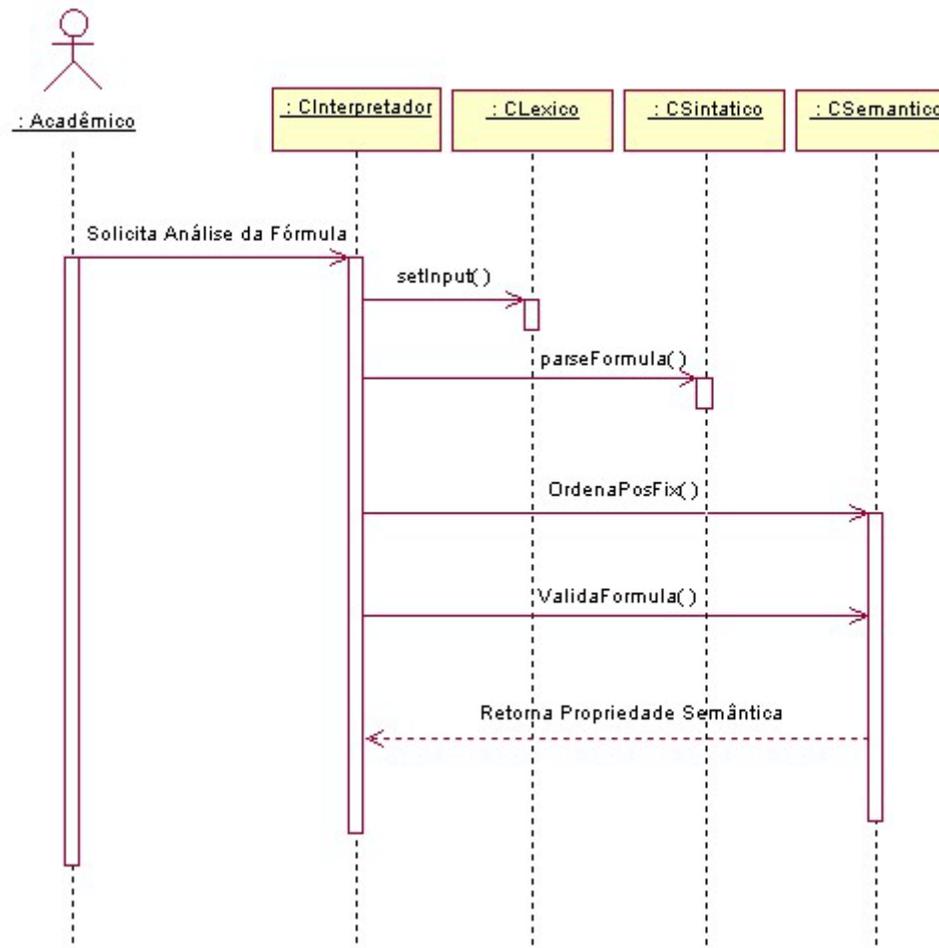
## Especificação da BNF do cálculo proposicional

```
<cálculo proposicional> ::= <argumento> | <fórmula>  
  
<argumento> ::= ( <premissas> ) -> <conclusão>  
<premissas> ::= <fórmula> | <fórmula> & <premissas>  
<conclusão> ::= <fórmula>  
  
<fórmula> ::= <símbolo proposicional>  
          | <símbolo verdade>  
          | ~ <fórmula>  
          | ( <fórmula> <conectivo> <fórmula> )
```

# Especificação da aplicação: diagrama de classes

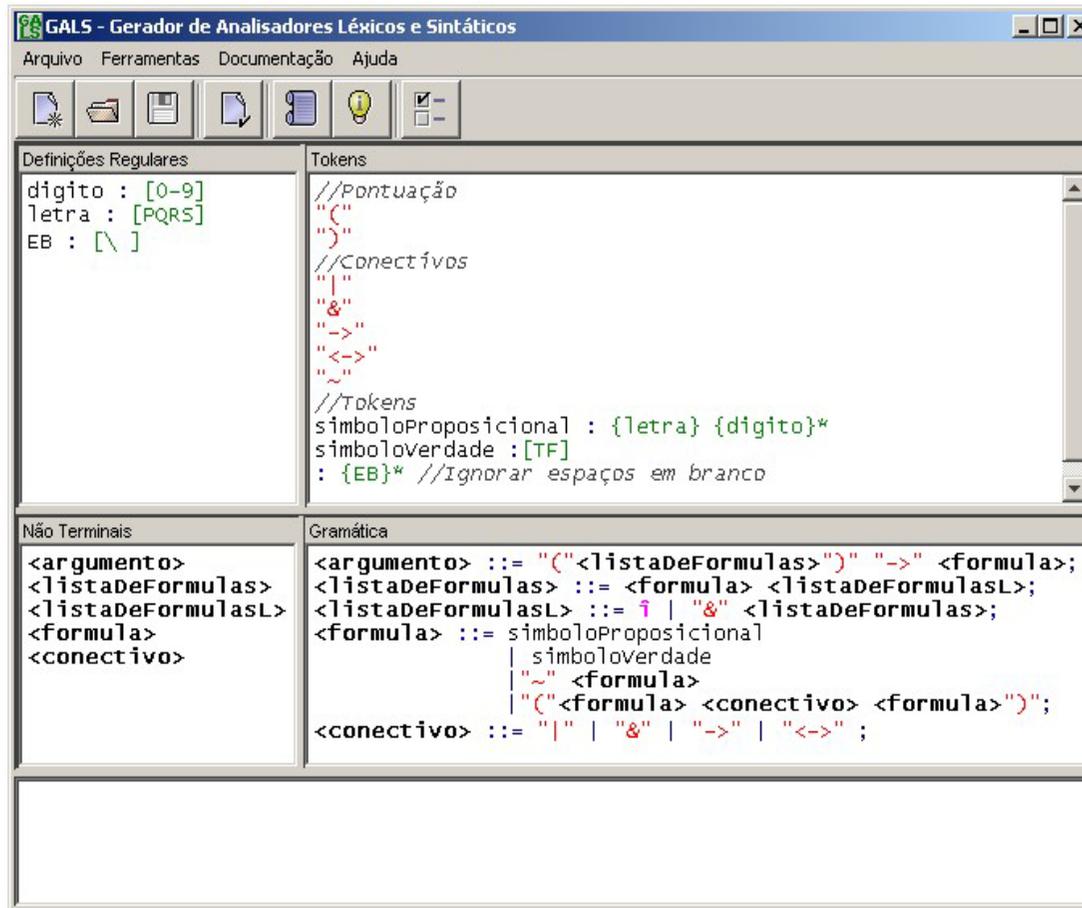


# Especificação da aplicação: diagrama de seqüência



# Implementação

## GALS



The screenshot shows the GALS (Gerador de Analisadores Léxicos e Sintáticos) application window. The title bar reads "GALS - Gerador de Analisadores Léxicos e Sintáticos". The menu bar includes "Arquivo", "Ferramentas", "Documentação", and "Ajuda". The toolbar contains icons for file operations and help. The main interface is divided into four panes:

- Definições Regulares:** Contains regular expressions for "dígito : [0-9]", "letra : [PQRS]", and "EB : [\\ ]".
- Tokens:** Lists tokens for punctuation ("(", ")", ",", "&", "-", "<->", "~"), connectives ("|", "&", "->", "<->"), and symbols. It also defines "símboloProposicional : {letra} {dígito}\*" and "símboloVerdade : [TF]".
- Não Terminais:** Lists non-terminals: "<argumento>", "<listaDeFormulas>", "<listaDeFormulasL>", "<formula>", and "<conectivo>".
- Gramática:** Defines grammar rules for the non-terminals, such as "<argumento> ::= \"(\"<listaDeFormulas>)\", \"->\" <formula>", "<listaDeFormulas> ::= <formula> <listaDeFormulasL>", and "<conectivo> ::= \"|\" | \"&\" | \"->\" | \"<->\" ;".

# Implementação: notação pós-fixada

$((P \vee Q) \rightarrow P)$

Pilha	Saída
(	
(	
(	P
$\vee$	P
$\vee$	PQ
(	$PQ \vee$
$\rightarrow$	$PQ \vee$
$\rightarrow$	$PQ \vee P$
	$PQ \vee P \rightarrow$

# Implementação: avaliação de uma fórmula na sua forma pós-fixada

Expressão	Elementos	Pilha
PQ $\vee$ P $\rightarrow$		
Q $\vee$ P $\rightarrow$	P	T
$\vee$ P $\rightarrow$	Q	F
P $\rightarrow$	$\vee$	T
$\rightarrow$	P	T
	$\rightarrow$	T

PQ  $\vee$  P  $\rightarrow$

Y = F  
X = T

Y = T  
X = T

# Operacionalidade

The image displays three screenshots of a logic interpreter window titled "Interpretador de fórmulas do cálculo...". Each window shows a menu bar with "Ferramenta" and "Sobre", a toolbar with icons for file operations and a "TIP" button, and a main area with a "Fórmula/Argumento" field, a "Resultado" field, and an "Erro" field. A "Fechar" button is located at the bottom right of each window.

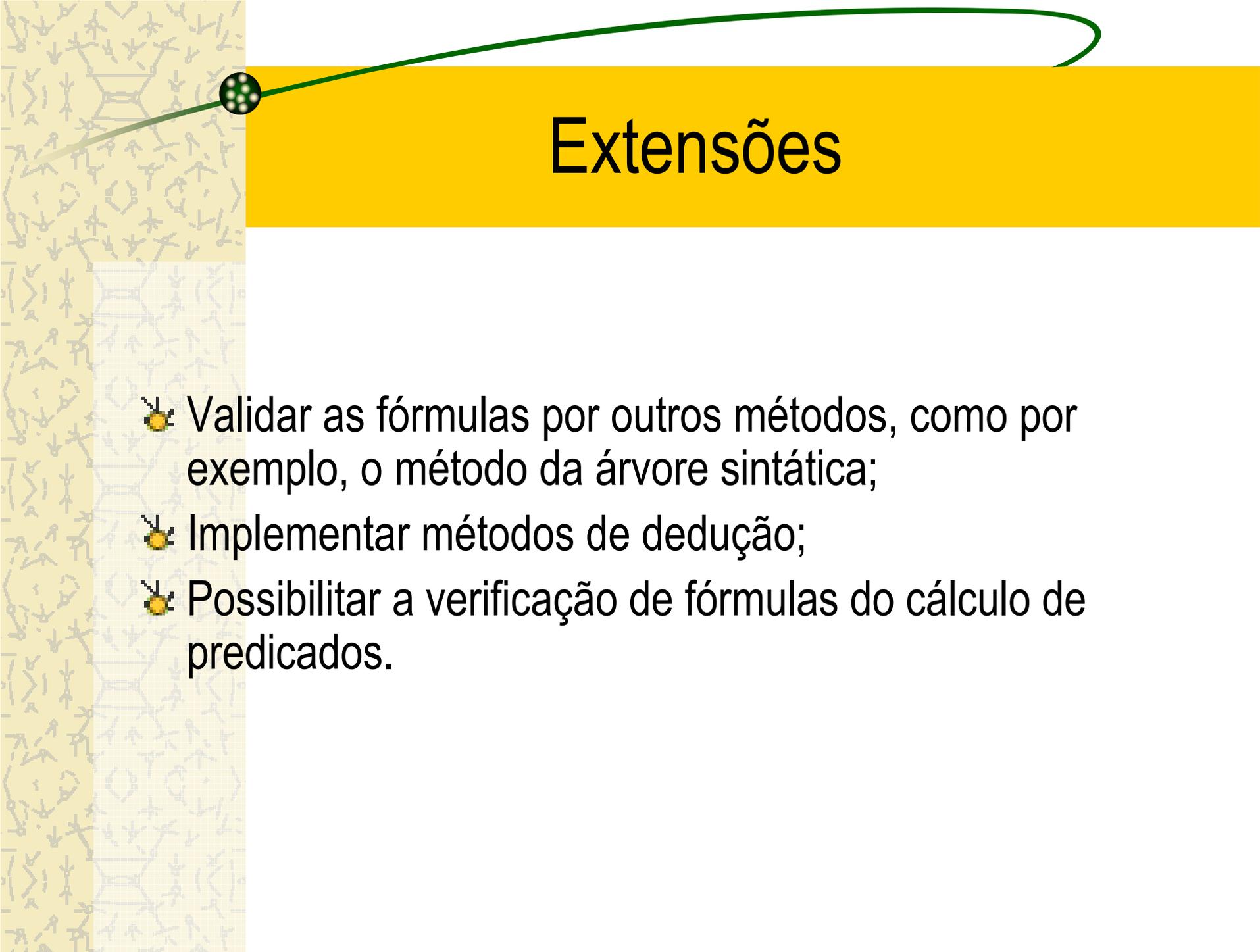
**Top Left Window:**  
Fórmula/Argumento:  $( (P \& Q) \rightarrow (P \leftrightarrow Q) )$   
Resultado: A fórmula  $( (P \& Q) \rightarrow (P \leftrightarrow Q) )$  é uma TAUTOLOGIA.

**Top Right Window:**  
Fórmula/Argumento:  $( \sim P \& (P \& \sim Q) )$   
Resultado: A fórmula  $( \sim P \& (P \& \sim Q) )$  é CONTRADITÓRIA.

**Bottom Center Window:**  
Fórmula/Argumento:  $( (P \mid Q) \rightarrow P )$   
Resultado: A fórmula  $( (P \mid Q) \rightarrow P )$  é SATISFATIVEL.

# Conclusão

- ✦ trabalho atingiu os objetivos propostos;
- ✦ foram estudados conceitos de lógica e cálculo proposicional, e princípios e técnicas de construção de compiladores;
- ✦ foram utilizadas três ferramentas:
  - ✦ *Rational Rose*;
  - ✦ Microsoft Visual C++;
  - ✦ GALS;



# Extensões

- ✦ Validar as fórmulas por outros métodos, como por exemplo, o método da árvore sintática;
- ✦ Implementar métodos de dedução;
- ✦ Possibilitar a verificação de fórmulas do cálculo de predicados.