Linguagem de Programação
Linguagem Lógica
Parte 2: Prova de Teoremas

Prof. Arnaldo Candido Junior

UNESP – IBILCE

São José do Rio Preto, SP

Lógica

- **Lógicas que estudaremos:**
  - Lógica Proposicional; de predicados (1ª ordem); fuzzy (ou difusa)
- **Outras lógicas:**
  - Lógica Modal (pode, precisa, não deveria, …)
  - Lógica Temporal (sempre, as vezes, eventualmente)
  - Parassindética, quântica

Lógica (2)

- Representação

operador	**símbolos**
conjunção (and)	∧ ∩ . &
dijunção (or)	∨ ∪ + |
negação (not)	~ ¬
implicação	→ ⊃
equivalência	≡ ↔

Lógica Proposicional

- Baseada em proposições, isto é declarações (afirmações, negações)
- Exemplo:
  p = hoje está chovendo
- Provar p equivale a provar que hoje está chovendo

Revisão

Conjunção

A	B	C=A∧B
0	0	0
0	1	0
1	0	0
1	1	1

Disjunção

A	B	C=A∨B
0	0	0
0	1	1
1	0	1
1	1	1

Negação

A	B=¬A
0	1
1	0

Disjunção Exclusiva

A	B	C=A ⊕ B
0	0	0
0	1	1
1	0	1
1	1	0

Implicação

A	B	C=A→B
0	0	1
0	1	1
1	0	0
1	1	1

Equivalência

A	B	C=A↔B
0	0	0
0	1	1
1	0	1
1	1	0

Equivalências

Propriedade	**Equivalência**
1. Idempotência	$A \rightarrow B \equiv \sim A \vee B$ **(importante)**
	$A \wedge \sim A \equiv .F.$ (paradoxo)
	$A \vee \sim A \equiv .V.$ (tautologia)
2. Comutativa	$A \wedge B \equiv B \wedge A$
	$A \vee B \equiv B \vee A$
3. Distributiva	$A \wedge (B \vee C) \equiv (A \wedge B) \vee (A \wedge C)$
	$A \vee (B \wedge C) \equiv (A \vee B) \wedge (A \vee C)$
4. Associativa	$A \wedge (B \wedge C) \equiv (A \wedge B) \wedge C$
	$A \vee (B \vee C) \equiv (A \vee B) \vee C$
5. Absorção	$A \vee (A \wedge B) \equiv A$
	$A \wedge (A \vee B) \equiv A$
6. De Morgan	$\sim(A \wedge B) \equiv \sim A \vee \sim B$
	$\sim(A \vee B) \equiv \sim A \wedge \sim B$

Análise Formal

- **Sintaxe**: tabelas verdade e regras para manipular os símbolos
  - Ex.: $p \wedge \sim p \equiv .F.$
- **Semântica**: significado dos símbolos
  - Ex.: p = hoje está chovendo

Língua Natural

- A implicação na lógica não é idêntica ao “if” da língua natural, o que pode ser contra-intuitivo. Considere:
  - p = a semana começa na terça → Brasília é a capital da Argentina
  - p é verdadeira! (.F. implica .F.)
  - Esse erro de raciocínio é conhecido como falácia da implicação material, afirmação do consequente ou confusão entre suficiência e necessidade

Implicação (2)

- Para um provador de teoremas, a implicação é a operação mais importante
- A seguir, veremos operações importantes para o raciocínio dedutivo
  - Modus Ponens
  - Modus Tolens
  - Transitividade da implicação

Modus Ponens

- **Se** p é verdadeiro
  e (p → q) também é verdadeiro
  então q é verdadeiro
- Exemplo:
  - p: ivo está nadando
  - p → q: quem está nadando está molhado
  - q: o que podemos concluir sobre ivo?

Modus Ponens (2)

p	q	p→q	p^(p→q)	[p^(p→q)]→q
F	F	V	F	V
F	V	V	F	V
V	F	F	F	V
V	V	V	V	V

Modus Tollens

- **Se** q é falso
  e (p → q) é verdadeiro
  então p é falso
- Exemplo:
  - p: o que podemos concluir sobre ivo?
  - p → q: quem está nadando está molhado
  - q: ivo não está molhado

Modus Tollens (2)

p	q	p→q	~q	(p→q)^~q	~p	[(p→q)^~q]→~p
F	F	V	V	V	V	V
F	V	V	F	F	V	V
V	F	F	V	F	F	V
V	V	V	F	F	F	V

Equivalência da Implicação

- Considere a equivalência:
  $p \rightarrow q \equiv \sim p \vee q$
- Exemplo:
  - Q: quem esta nadando está molhado
  - R: alguém não está nada ou estará molhado

Transitividade da implicação

- Quando $p \rightarrow q$ e $q \rightarrow r$ são válida, podemos concluir que $p \rightarrow r$ também é válida
- Demonstração a seguir

Transitividade da implicação (2)

p	q	r	p→q	q→r	(p→q)^(q→r)	p→r	[(p→q)^(q→r)]→(p→r)
F	F	F	T	T	T	T	T
F	F	T	T	T	T	T	T
F	T	F	T	F	F	T	T
F	T	T	T	T	T	T	T
T	F	F	F	T	F	F	T
T	F	T	F	T	F	T	T
T	T	F	T	F	F	F	T
T	T	T	T	T	T	T	T

Limitações

- A lógica proposicional possui algumas limitações para uso em provadores de teoremas
- Exemplo: dificuldade de expressar quantificadores
  - **Todo aquele** que nada está molhado
  - **Todos** os humanos são mortais
  - **Alguém** está nadando

Lógica de Predicados

- Baseada no trabalho de Gottfried Frege
- Subdiviões
  - Primeira ordem (**foco**)
  - Segunda ordem
  - Outros casos

Lógica de Predicados (2)

- Contém predicados (functores) e constantes (argumentos dos predicados)
  - Predicados tem a ver com **relações**
  - Argumentos tem a ver com **objetos**
- Predicado não é uma caixa preta como uma proposição na lógica proposicional

Lógica de Predicados (3)

- Traz o conceito de variáveis (como visto em Prolog)
- Traz o conceito de quantificadores: universal (∀ - qualquer) e existencial (∃ - existe)

Exemplos

- **Predicados** e **constantes**: vizinho(joao, maria)
  - João é vizinho de Maria
- **Variáveis**: vizinho(joao, X)
  - João é vizinho de X
- **Funções**: pai(joão) = antônio
  - O pai de João é Antônio

Exemplos (2)

- **Quantificador universal**:
  $\forall_C$ [criança(C) → gosta(C, sorvete)]
- Qualquer que seja C tal que se C é uma criança então C gosta de sorvete
- Ou ainda: todas as crianças gostam de sorvete

Exemplos (2)

- **Quantificador existencial**:
  $\exists_O$[oceano(O) → banha(O, Brasil)]
- Existe O tal que se O é um oceano então O banha o Brasil
- Ou ainda: há um oceano que banha o Brasil

Negação dos Quantificadores

- A lógica de predicados nem sempre é intuitiva na língua natural
- Convém observar que a negação dos quantificadores funciona da seguinte forma:
  - A negação de $\forall_x$ **[p(x)]** é $\exists_x$**[~p(x)]**
  - A negação de $\exists_x$**[p(x)]** é $\forall_x$ **[~p(x)]**

Inferência

- **Dedução** (**foco**): esses feijões são daquele pacote e todos os grãos daquele pacote são brancos
  - Logo, esses feijões são brancos
- **Indução**: Esses feijões são brancos e são daquele pacote
  - Logo, todos os grãos daquele pacote são brancos
- **Abdução**: Esses feijões são brancos e todos os grãos daquele pacote são brancos.
  - Logo: esses grãos são daquele pacote

Provador de Teoremas

- A seguir, veremos como construir um provador de teoremas simples, que manipula apenas valores booleanos
  - Usando uma estratégia chamada **princípio da resolução**
  - Começaremos com lógica proposicional e estenderemos o raciocínio para lógica de predicados (1ª ordem)

Princípio da Resolução

- Princípio da resolução é uma estratégia de inferência com a qual se tenta provar que a negação do objetivo
- É uma prova por contradição
- Se o objetivo é provar a proposição O, vamos assumir que ~O é verdadeiro

Princípio da Resolução (2)

- E verificar se isso leva a algum paradoxo na base de fatos
- Vamos assumir que a base de fatos é **correta** (sound)
  - Isto é, não contém inconsistência / paradoxos
- Com uma base correta, o algoritmo também será correto
  - Isto, deriva conhecimento válido

Princípio da Resolução (3)

- Para aplicar o algoritmo, precisamos converter a base na **forma clausal** (versão simplificada). Seguiremos o roteiro a seguir:
- Remover da base tautologias e paradoxos
- Aplicar distributiva da conjunção sempre que possível
- Continua …

Princípio da Resolução (4)

- Quebrar proposições com conjunção em cláusula menores
- Trocar implicação pela expressão disjunção equivalente
- Resultado final: apenas disjunções e negações em cada cláusula
- Vamos usar as regras a seguir...

Forma Clausal simplificada

- 1. Trocar **p ∧ ¬p** por **.F.**
- 2. Trocar **p ∨ ¬p** por **.V.**
- 3. Trocar **p ∨ (q ∧ r)** por **(p ∨ q) ∧ (q ∨ r)**
- 4. Quebrar **p ∧ q** em duas cláusulas menores **p** e **q**
- 5. Trocar **p → q** por **~p ∨ q**
- 6. Repetir até não houver mais regras a se aplicar

Exemplo: forma clausal simplificada

- Original:
  - 1. $(p \vee q) \wedge (\sim q \vee r) \wedge \sim s$
- Ajustado:
  - 2. $p \vee q$
  - 3. $\sim q \vee r$
  - 4. $\sim s$

Exemplo: forma clausal simplificada (2)

- Original:
  - 1. $p \vee (\sim q \wedge r \wedge s \rightarrow t)$
- Ajustado:
  - 2. $p \vee \sim q$
  - 3. $p \vee r$
  - 4. $p \vee \sim s \vee t$ (vem de $p \vee (s \rightarrow t)$)

Algoritmo

- No slide 31, 6 regras foram criadas para colocar a base na forma
- Agora usaremos três regras adicionais para fazer o processo de inferência
  - 7. Aplicar modus ponens
  - 8. Aplicar modus tollens
  - 9. Aplicar transitividade da implicação

Algoritmo (2)

- Regra 7: modus ponens
  - Na base:
    - 1. p
    - 2. ~p v q (equivalente de p → q)
  - Inferido:
    - 3. q

Algoritmo (3)

- Regra 8: modus tollens
  - Na base:
    - 1. ~p ∨ q (equivalente de p → q)
    - 2. ~q
  - Inferido:
    - 3. ~p

Algoritmo (4)

- Regra 9: transitividade da implicação
  - Na base:
    - 1. ~p v q (equivalente de p → q)
    - 2. ~q v r (equivalente de q → r)
  - Inferido:
    - 3. ~p v r (equivalente de p → r)

Algoritmo (5)

- Regra informal: p e ~p se anulam sempre que estão em regras diferentes
  - Versão intuitiva das regras 7 a 9
  - Regras são simplificadas e outros casos mais complexos são contemplados também

Exemplo: aspirina

- Fatos:
  - 1. p: temperatura do paciente > 38.2
  - 2. p → q: quem está com temperatura > 38.2, tem febre
  - 3. q → r: quem tem febre precisa de aspirina
- Desejamos provar r (o paciente precisa de aspirina). Inserir ~r na base

Exemplo: aspirina (2)

- Passo 0: passar para forma clausal simplificada
  - 1. p
  - 2. ~p v q
  - 3. ~q v r

Exemplo: aspirina (3)

- Passo 1: inserir ~r na base (prova por contradição)
  - 1. p
  - 2. ~p v q
  - 3. ~q v r
  - **4. ~r**

Exemplo: aspirina (4)

- Passo 2: combinar 3 e 4 usando a regra informal
  - 1. p
  - 2. ~p v q
  - **3. ~q v r**
  - **4. ~r**
  - 5. ~q

Exemplo: aspirina (5)

- Passo 3: combinar 2 e 5
  - 1. p
  - **2. ~p ∨ q**
  - 3. ~q v r
  - 4. ~r
  - **5. ~q**
  - 6. ~p

Exemplo: aspirina (6)

- Passo 4: combinar 1 e 6
  - **1. p**
  - (...)
  - **6. ~p**
  - 7. Absurdo: fatos 1 e 6 não podem ser verdadeiros ao mesmo tempo

Exemplo: aspirina (7)

- Paradoxo ou contradição detectados!
- O paradoxo se dá porque existe uma informação incorreta na base
- Assumimos que ~r era verdade (não receitar aspirina)
- A única explicação possível é que ~r é falso. Assim, r é verdade (receitar aspirina)

Ordem de resolução

- A vantagem da prova por contradição é que existem vários caminhos a se buscar:
  - Exemplo visto: começou combinando 3 e 4 e seguiu daí.
  - Outro exemplo: começar combinando 1 e 2 e usar os fatos inferidos a partir daí
  - Múltiplos caminhos até a solução otimiza o desempenho do algoritmo estudado

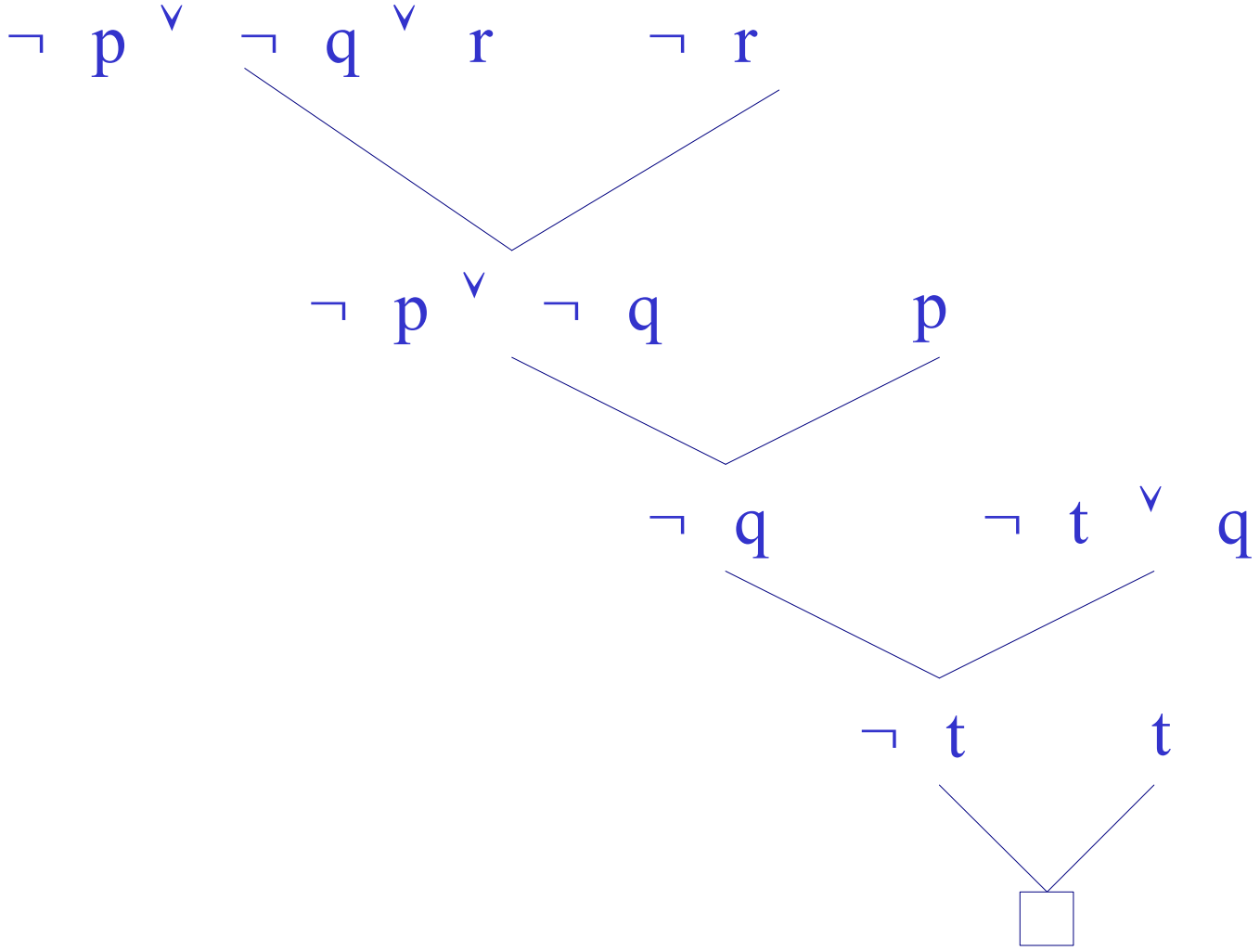
Resolução: segundo exemplo

- Exercício 1: passar para a forma clausal simplificada
  - 1. p
  - 2. $(p \wedge q) \rightarrow r$
  - 3. $(s \vee t) \rightarrow q$
  - 4. t

Resolução: segundo exemplo (2)

- Solução:
  - 1. p
  - 2. ~p v ~q v r (antigo 2)
  - 3. ~s v q (antigo 3)
  - 4. ~t v q (antigo 3)
  - 5. t (antigo 4)
- Exercício 2: provar que r é verdadeiro (fato 6 = ???)

Resolução: segundo exemplo (3)



Resolução: segundo exemplo (4)

- Note que existem becos sem saída na resolução
  - Caso encontre um, recomeçar análise
- Prolog reduz chances de becos sem saída começando a inferência pela contradição
  - Isto é, o último fato adicionado a base
  - Esse processo é conhecido como busca com encadeamento para trás

Resolução: segundo exemplo (2)

- Obs 2: o exemplo dado pode ser representado por uma árvore
  - Em situações reais, trata-se de uma floresta
  - Para simplificar os exemplos, vamos usar árvores

Princípio da resolução: lógica de predicados

- Na lógica proposicional é mais simples fazer uma prova por contradição
  - **p** e **~p** são contraditórios
- Na lógica de predicados, o equivalente seria:
  - **brasileiro(joao)** e **~brasileiro(joao)**
- Note que **brasileiro(joao)** e **~brasileiro(jose)** não são contraditórios

Unificação

- Variáveis: fator complicador para aplicar o princípio da resolução em lógica de 1ª ordem
- Estratégia: **unificação**
  - Atribuir um valor a uma variável que leve a um paradoxo de interesse
- Exemplo: **brasileiro(X) ∧ ~brasileiro(jose)**
  - Se instanciarmos **X=jose**, então temos o paradoxo

Unificação (2)

- Veremos a unificação em um exemplo completo
  - Modelando fatos do mundo real
  - Convertendo para lógica de 1ª ordem
  - Convertendo para fora clausal
  - Derivando fatos e unificando variáveis

Exemplo completo

- Parte 1: converta os seguintes fatos para lógica de 1ª ordem
  - Os gatos são mamíferos
  - Todo o mamífero tem um progenitor
  - O filho de um gato é um gato
  - Teco é um gato
  - Teco é progenitor do Navalha

Exemplo completo (2)

- Os gatos são mamíferos: $\forall_G$**[gato(G) → mamifero(G)]**
- Todo o mamífero tem um progenitor: $\forall_G \exists_P$**[mamifero(G) → progenitor(P,G)]**
- O filho de um gato é um gato: $\forall_{G,F}$**[gato(G) ∧ progenitor(G, F) → gato(F)]**
- Teco é um gato: **gato(teco)**
- Teco é progenitor do Navalha: **progenitor(teco, navalha)**

Exemplo completo (3)

- Importante: existem formas diferentes de se “traduzir” conhecimento do mundo real para o formalismo lógico
  - De acordo com a modelagem, pode ser mais fácil ou mais difícil resolver um problema
  - Uma boa modelagem depende da experiência e intuição do projetista

Exemplo completo (4)

- Parte 2: remova os operadores existencial e universal para formatar a base
- Parte 3: passe a base para a forma clausal
- Parte 4: prove que Navalha é um gato

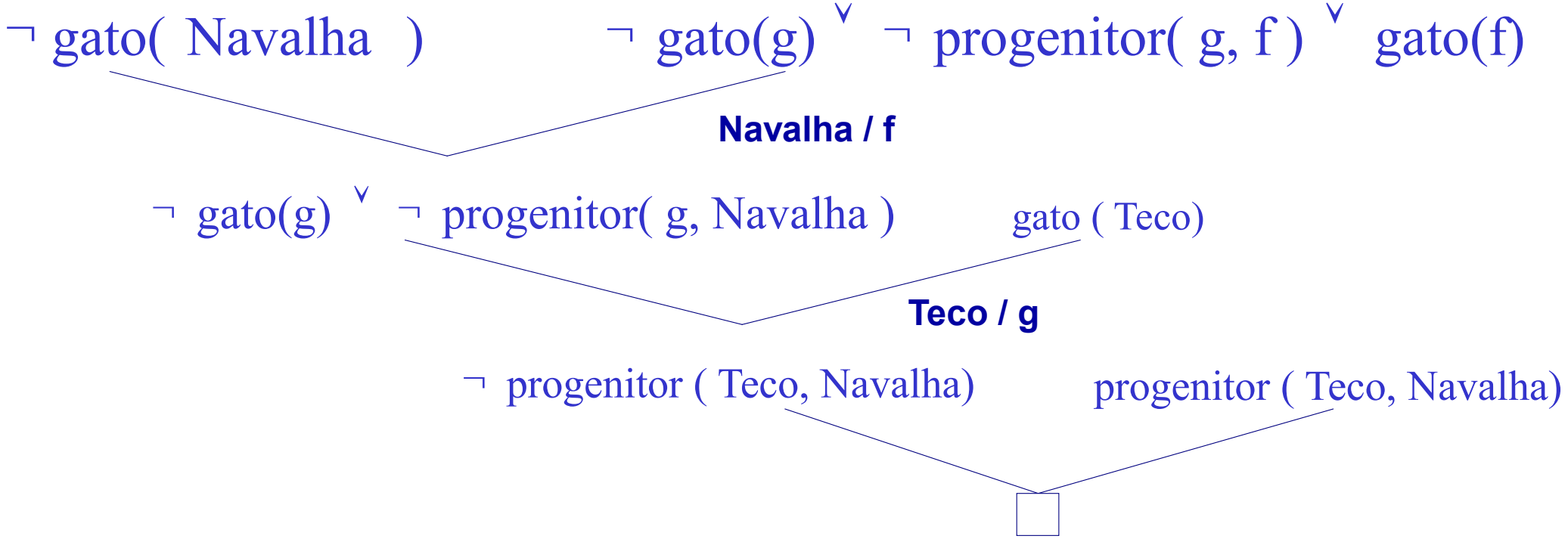
Exemplo completo (5)

- Parte 2: sem quantificadores
  - 1. gato(G) → mamifero(G)
  - 2. mamifero(G) → progenitor(P, G)
  - 3. (gato(G) ∧ progenitor(G, F)) → gato(F)
  - 4. gato(teco)
  - 5. progenitor(teco, navalha)

Exemplo completo (6)

- Parte 3: forma clausal simplificada
  - 1. ¬gato(G) ∨ mamifero(G)
  - 2. ¬mamifero(G) ∨ progenitor(P, G)
  - 3. ¬gato(g) v¬ progenitor(G, F) ∨ gato(F)
  - 4. gato(teco)
  - 5. progenitor(teco, navalha)
  - 6. ¬gato(navalha)

Exemplo completo (7)



Modelagem

- Considere o texto
  - “Os gatos gostam de peixe. Os gatos comem tudo do que gostam. Teco é um gato”
- Passar para lógica de 1ª ordem
  - Note que existem várias formas de se representar
  - Exploraremos uma específica

Modelagem (2)

- Os gatos gostam de peixe
  $\forall_G$[gato(G) → gosta(G, peixe)]
- Os gatos comem tudo do que gostam
  ∀G[gosta(G, C) → come (G, C)]
- Teco é um gato
  gato(teco)
- Provar que teco como peixe

Exercícios (2)

- Parte 1: represente as afirmações seguintes usando lógica de predicados de 1ª ordem
  - Um animal pesado come muito
  - Os elefantes são animais grandes
  - Todos os elefantes têm um alimento preferido
  - Dumbo é um elefante
  - Todos os animais grandes são pesados
  - O amendoim é um alimento

Exercícios (3)

- $\forall_A$ [animal(A, pesado) $\rightarrow$ come(A,muito)]
- $\forall_A$ [elefante(A) $\rightarrow$ animal(A, grande)]
- $\forall_A \exists_C$ [elefante(A) $\rightarrow$ alimento(C, preferido)]
- elefante(dumbo)
- $\forall$A [animal(A, grande) $\rightarrow$ animal(A, pesado)]

Exercícios (4)

- Parte 2: remova os operadores existencial e universal para formatar a base
- Parte 3: passe a base para a forma clausal simplifcada
- Parte 4: prove que Dumbo come muito

Créditos

- Parcialmente adaptado dos slides do professor:
  - Carlos Fernando da Silva Ramos (IPP)