Inteligência Artificial
RC&R em Lógica

Jerusa Marchi

jerusa.marchi@ufsc.br

Departamento de Informática e Estatística

Universidade Federal de Santa Catarina

Florianópolis - SC

Lógica

- A Lógica trata de dois conceitos: Verdade e Prova
- Sistema lógico:
  - conjunto de fórmulas que podem assumir valores verdadeiro (V) ou falso (F)
    - fórmula válida - uma fórmula pode ser válida se há uma atribuição de valores verdade que a faça verdadeira
    - tautologia - a fórmula que é **sempre** verdadeira
    - Teoria dos Modelos
  - conjunto de regras de inferência
    - quando aplicada repetidamente a fórmulas verdadeiras, gera novas fórmulas verdadeiras (dedução)
    - as fórmulas geradas constituem uma prova
    - Teoria das Provas

Sintaxe

- *Linguagem lógica de primeira ordem*: $L(\mathbf{P}, \mathbf{F}, \mathbf{C}, \mathbf{V})$, onde:
  - Um conjunto $\mathbf{P}$ de *símbolos de predicado*
  - Um conjunto $\mathbf{F}$ de *símbolos de função*
  - Um conjunto $\mathbf{C}$ de *símbolos de constante*
  - Um conjunto $\mathbf{V}$ de *símbolos de variável*
- Aridade: a cada símbolo de predicado ou função é associado um número de argumentos.
  - Se a linguagem possuir somente símbolos de predicado com aridade zero, a lógica é chamada de *proposicional* e os conjuntos $\mathbf{F}$, $\mathbf{C}$ e $\mathbf{V}$ não são importantes para a definição da linguagem

Sintaxe

- A sintaxe da linguagem pode ser definida através da seguinte linguagem formal:

<fórmula>::=<fórmula-atômica> | $\neg$ <fórmula> | | (<fórmula>) |
<fórmula> $\wedge$ <fórmula> |
<fórmula> $\vee$ <fórmula> |
<fórmula>$\rightarrow$<fórmula> |
$\forall$ <variável> .(<fórmula>) |
$\exists$ <variável> .(<fórmula>)

<fórmula-atômica>::=V | F |
<predicado> (<termo>, . . . , <termo>)

<termo>::=<variável> | <constante> |
<função> (<termo>, . . . , <termo>)

Sintaxe

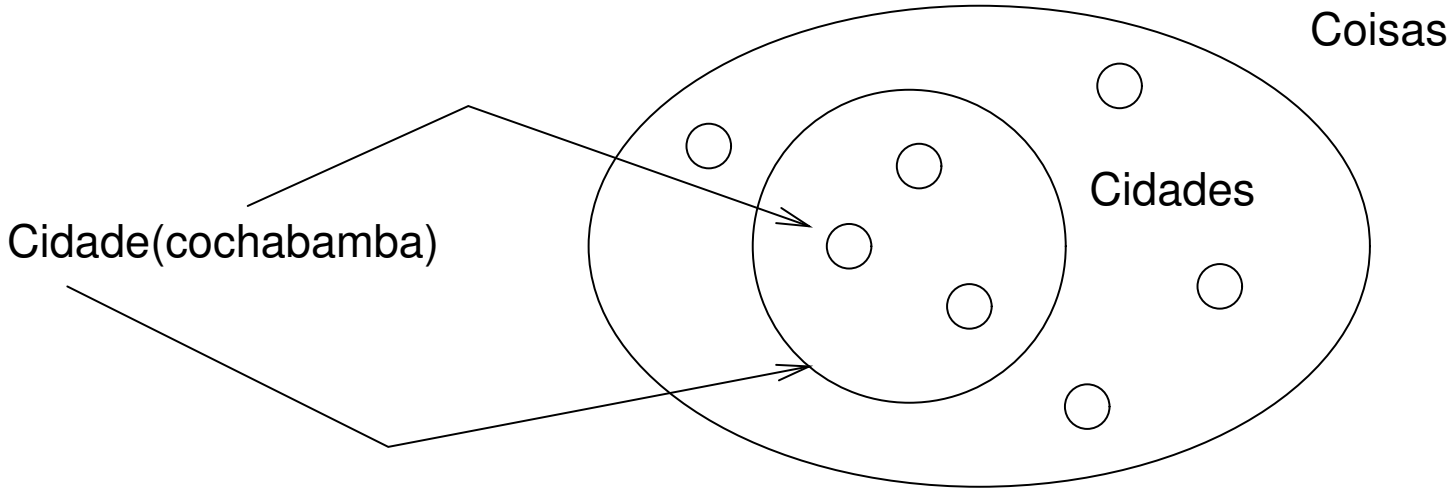
<constante>::= a, b, c, d...
<variável>::= x, y, z, w...
<função>::= f, g, h...
<predicado>::= P, Q, R...

- Exemplos de fórmulas:

$$(P \vee Q) \rightarrow S \qquad \neg(\neg P \vee \neg Q \rightarrow R),$$
$$((Pai(a,b) \wedge Pai(b,c)) \rightarrow Avo(a,c)), \qquad \neg(Ama(brutus, cesar)),$$
$$\forall x.((P(x) \wedge \neg(P(a))) \rightarrow Q(b)), \qquad \forall x.\exists y.(Gosta(y,x)),$$
$$Ama(amelia, z), \qquad F.$$

Semântica

- Tarski, 1956 - Formalização de como atribuir semântica (significado) à linguagem lógica
- Associar os elementos sintáticos da linguagem a estruturas matemáticas da teoria de conjuntos
  - Linguagem natural:
    - Cochabamba é uma Cidade.
  - Lógica:



Semântica

- Interpretação:
  - Um domínio $\mathcal{D}$, um conjunto não vazio.
  - Uma função de associação ξ que leva cada elemento da sintaxe da linguagem em uma estrutura matemática definida sobre o conjunto $\mathcal{D}$.

Semântica

- A função de associação ξ é definida da seguinte maneira:
  - Cada símbolo de constante é associado a um elemento de $\mathcal{D}$
  - Cada símbolo de função de aridade n é associado a uma função de $\mathcal{D}^n$ em $\mathcal{D}$, onde $\mathcal{D}^n$ representa o produto cartesiano do conjunto $\mathcal{D}$ com ele mesmo n vezes
  - Cada símbolo de predicado de aridade n é associado a uma relação de aridade n contida em $\mathcal{D}^n$
- Extensão da função de associação para termos:
  - Se $f(t_1, \ldots, t_n)$ é um termo, então:

$$\xi(f(t_1, \ldots, t_n)) = \xi(f)(\xi(t_1), \ldots, \xi(t_n)).$$

Semântica

- **Valor verdade** de uma fórmula lógica:
  - $P(t_1, \ldots, t_n)$ tem valor verdade *V* se e somente se $(\xi(t_1), \ldots, \xi(t_n)) \in \xi(P)$.
  - $\neg A$ tem valor verdade *V* se e somente se *A* tem valor verdade *F*.
  - $A \wedge B$ tem valor verdade *V* se e somente se *A* e *B* tem valor verdade *V*.
  - $A \vee B$ tem valor verdade *V* se e somente se *A* ou *B*, ou ambos, têm valor verdade *V*.
  - $A \rightarrow B$ tem valor verdade *V* se e somente se $(\neg A \vee B)$ tem valor verdade *V*.
  - $\forall x.A$ tem valor verdade *V* se e somente se para todo $\alpha \in \mathcal{D}$, se $\xi(t) = \alpha$, onde t é um termo, então $A\{x/t\}$ tem valor verdade *V*.
  - $\exists x.A$ tem valor verdade *V* se e somente se existe um $\alpha \in \mathcal{D}$, tal que $\xi(t) = \alpha$, onde t é um termo, e $A\{x/t\}$ tem valor verdade *V*.

Semântica

- Tabelas verdade:

A	B	$A \wedge B$	$A \vee B$	$A \rightarrow B$	$\neg A$
F	F	F	F	V	V
F	V	F	V	V	V
V	F	F	V	F	F
V	V	V	V	V	F

- Interpretação: possíveis combinações de valores verdades
- Modelo: interpretação que tem valor verdade verdadeiro
- Se existe um modelo então diz-se que a fórmula é válida
- Se todas as interpretações da fórmula são modelos, diz-se que a fórmula é uma tautologia

Semântica

- Equivalências válidas:

$$(A \vee B) \Leftrightarrow \neg(\neg A \wedge \neg B)$$

$$(A \wedge B) \Leftrightarrow \neg(\neg A \vee \neg B)$$

$$(A \rightarrow B) \Leftrightarrow (\neg A \vee B)$$

$$A \vee (B \wedge C) \Leftrightarrow (A \vee B) \wedge (A \vee C)$$

$$A \wedge (B \vee C) \Leftrightarrow (A \wedge B) \vee (A \wedge C)$$

$$\forall x.(A) \Leftrightarrow \neg\exists x.\neg(A)$$

Semântica

- Exemplo: Considere a seguinte linguagem lógica $L(\mathbf{P}, \mathbf{F}, \mathbf{C}, \mathbf{V})$ com:
  - $\mathbf{P} = \{P, Q\}$
  - $\mathbf{F} = \{f, g\}$
  - $\mathbf{C} = \{a\}$
  - $\mathbf{V} = \{x\}$

  e as seguintes fórmulas lógicas:

$$P(a)$$

$$\forall x.(P(x) \rightarrow Q(x))$$

$$\forall x.(P(x) \rightarrow P(f(x)))$$

$$\forall x.(P(x) \rightarrow Q(g(x))))$$

Semântica

- Interpretação:
  - Domínio: $\mathcal{D} = \mathbb{Z}$, conjunto dos números inteiros
  - Função de associação:

x	P	Q	f	g	a
$\xi(x)$	$\mathbb{N}$	$\mathbb{Z}$	$\lambda x.x + 1$	$\lambda x. - x$	0

- Agora as fórmulas da linguagem ganham o seguinte significado:
  - 0 é um número natural
  - Todo número natural é um inteiro
  - Todo sucessor de um número natural é também um natural
  - Todo oposto de um número natural é um inteiro

RC em Lógica

- Representação de Conhecimento em Lógica: Conhecimento Declarativo
  - Declaração de Fatos
  - Declaração de Regras

RC em Lógica

1. Marcos era um homem

$$homem(marcos)$$

2. Marcos nasceu em Pompéia

$$pompeano(marcos)$$

3. Todos os que nasceram em Pompéia eram romanos

$$\forall x.pompeano(x) \rightarrow romano(x)$$

4. César era um soberano

$$soberano(cesar)$$

RC em Lógica

5. Todos os romanos eram leais a César ou então odiavam-no

$$\forall x.romano(x) \rightarrow leal(x, cesar) \vee odeia(x, cesar)$$

6. Todo o mundo é leal a alguém

$$\forall x.\exists y.leal(x, y)$$

7. Os homens só tentam assassinar soberanos aos quais não são leais

$$\forall x.\forall y.homem(x) \wedge soberano(y) \wedge tentaAssassinar(x, y) \rightarrow \neg leal(x, y)$$

8. Marcos tentou assassinar César

$$tentaAssassinar(marcos, cesar)$$

Raciocínio em Lógica

- Dado um conjunto de fatos e regras, inferir novos conhecimentos que sejam verdadeiros
- Um processo de inferência consiste na aplicação de *regras de inferência* corretas sobre o conjunto de fórmulas, produzindo novas fórmulas válidas
- Dado um conjunto G de fórmulas e uma fórmula W, determinar se W é ou não uma consequência lógica de G

$$G \models W$$

Raciocínio em Lógica

- Algumas regras de inferência conhecidas

$(A \wedge (A \rightarrow B)) \rightarrow B$	*Modus Ponens*
$(\neg B \wedge (A \rightarrow B)) \rightarrow \neg A$	*Modus Tollens*
$((A \rightarrow B) \wedge (B \rightarrow C)) \rightarrow (A \rightarrow C)$	*Silogismo Hipotético*
$\forall x.A \rightarrow A\{x/a\}$	*Especialização*
$A\{x/a\} \rightarrow \exists x.A$	*Generalização*

Raciocínio em Lógica

Marcos era leal a Cesar?

- Fatos

$$homem(marcos), soberano(cesar), tentaAssassinar(marcos, cesar)$$

  - Regra

$$\forall x.\forall y.homem(x) \wedge soberano(y) \wedge tentaAssassinar(x, y) \rightarrow \neg leal(x, y)$$

  - Aplicando Modus Ponens

$$(A \wedge (A \rightarrow B)) \rightarrow B$$

  - Conclui-se que

$$\neg leal(marcos, cesar)$$

Raciocínio em Lógica

- Como realizar o raciocínio de forma automática?
  - Métodos Automáticos de Prova de Teoremas
    - Padronizar as fórmulas lógicas - CNF
    - Aplicar um método de Prova Automática
      - Método da Resolução
      - Método de Tableaux

Formas normais canônicas

- Podemos representar as fórmulas em uma forma normalizada utilizando unicamente os operadores $\{\neg, \vee, \wedge\}$
- Objetivo das formas normais:
  - Simplificar fórmulas complexas
  - Em geral, primeira etapa dos procedimentos de demonstração automática de teoremas

Formas normais canônicas

- Forma normal conjuntiva (ou forma clausal) - CNF
  - conjunção de disjunções

$$(p \vee q) \wedge (\neg q \vee r)$$

  - Mais formalmente
    - a fórmula está na forma

$$C_1 \wedge C_2 \wedge \ldots C_n$$

    - onde cada cláusula C_i é uma disjunção de literais

$$L_1 \vee L_2 \vee \ldots L_n$$

    - onde cada literal L_i é um símbolo de predicado ou sua negação

Formas normais canônicas

- Forma normal disjuntiva (ou forma clausal dual) - DNF
  - disjunção de conjunções

$$(r \wedge \neg p) \vee (q \wedge r)$$

  - Mais formalmente
    - a fórmula esta na forma

$$D_1 \vee D_2 \vee \ldots D_n$$

    - onde cada cláusula D_i é uma conjunção de literais

$$L_1 \wedge L_2 \wedge \ldots L_n$$

Formas normais canônicas

- As fórmulas são transformadas utilizando as relações de equivalência

$$A \rightarrow B \equiv \neg A \vee B \text{ (eliminação da implicação)}$$

$$\neg(A \vee B) \equiv \neg A \wedge \neg B \text{ (lei de De Morgan)}$$

$$\neg(A \wedge B) \equiv \neg A \vee \neg B \text{ (lei de De Morgan dual)}$$

$$A \vee (B \wedge C) \Leftrightarrow (A \vee B) \wedge (A \vee C) \text{ (distributividade)}$$

$$A \wedge (B \vee C) \Leftrightarrow (A \wedge B) \vee (A \wedge C) \text{ (distributividade)}$$

- As representações em formas normais são equivalentes às fórmulas originais

$$W \equiv CNF_W \equiv DNF_W$$

Algoritmo FNC

- Eliminar todas as ocorrências de $A \rightarrow B$ em W, substituindo-as por $\neg A \vee B$.
- Reduzir o escopo das negações de maneira que só restem negações aplicadas a fórmulas atômicas. Para isto usar as regras:

$$\neg(A \vee B) \Rightarrow (\neg A \wedge \neg B),$$
$$\neg(A \wedge B) \Rightarrow (\neg A \vee \neg B),$$
$$\neg(\forall x.A) \Rightarrow \exists x.\neg(A),$$
$$\neg(\exists x.A) \Rightarrow \forall x.\neg(A),$$
$$\neg(\neg(A)) \Rightarrow A.$$

Algoritmo FNC

- Substituir os nomes de variáveis de maneira que cada quantificador possua a sua própria variável.
- Mover os quantificadores, preservando sua ordem, para o início da fórmula.
- Eliminar os quantificadores existenciais. Este passo é realizado através do processo chamado *Skolemização*, criado por Skolem em 1920. A idéia básica é que uma fórmula com uma variável quantificada existencialmente se torna verdadeira quando esta variável é substituída por pelo menos um elemento do domínio. Como a existência deste elemento está garantida, podemos atribuir-lhe um *nome*, isto é, um símbolo de constante que não apareça na fórmula W:

  $\exists x.P(x) \Rightarrow P(a)$, onde a é um símbolo de constante, chamada *constante de Skolem*, que não aparece em W.

Algoritmo FNC

- Quando a variável quantificada existencialmente aparece dentro do escopo de um quantificador universal, o elemento do domínio associado ao quantificador existencial pode depender do valor escolhido para a variável quantificada universalmente, que pode ser qualquer. Por exemplo, na fórmula $\forall x.\exists y.(M\tilde{a}e(y,x))$, se adotarmos a interpretação "para qualquer *x* existe um *y* tal que *y* é mãe de *x*" fica claro que o valor de *y*, a mãe, depende do valor de *x*, o filho. Neste caso, é necessário substituir a variável quantificada existencialmente por um símbolo de função, cujos argumentos são todas as variáveis quantificadas universalmente que dominam o quantificador existencial:

  $\forall x_1.\ldots\forall x_n.\exists y.(P(y)) \Rightarrow P(f(x_1,\ldots,x_n))$, onde f é um símbolo de função, chamada *função de Skolem*, que não aparece em W

Algoritmo FNC

- Eliminar os quantificadores universais, deixando implícito que todas as variáveis que aparecem na fórmula são quantificadas universalmente.
- Converter a fórmula para a forma de uma conjunção de disjunções usando a propriedade distributiva do operador $\vee$ sobre o operador $\wedge$:

$$A \vee (B \wedge C) \Rightarrow (A \vee B) \wedge (A \vee C).$$

- Trocar os nomes das variáveis de maneira que cada cláusula do resultado possua suas variáveis próprias. Isto é possível devido ao resultado:

$$\forall x.(P(x) \wedge Q(x)) \Leftrightarrow \forall x.(P(x)) \wedge \forall y.(Q(y)).$$

Forma normal conjuntiva

- Exemplo:
  - "Se chove então eu não saio. Mas, se não chove, eu saio e eu tomo sorvete. Independentemente, eu saio e tomo sorvete."

representamos como:	transformando em CNF:
$c \rightarrow \neg s$	$\neg c \vee \neg s$
$\neg c \rightarrow s \wedge t$	$c \vee s$
$s \wedge t$	$c \vee t$
	s
	t

Forma normal conjuntiva

- Todos os que nasceram em Pompéia eram romanos

$$\forall x.pompeano(x) \rightarrow romano(x)$$

$$\neg pompeano(x_1) \vee romano(x_1)$$

- Todos os romanos eram leais a César ou então odiavam-no

$$\forall x.romano(x) \rightarrow leal(x, cesar) \vee odeia(x, cesar)$$

$$\neg romano(x_2) \vee leal(x_2, cesar) \vee odeia(x_2, cesar)$$

Forma normal conjuntiva

- Todo o mundo é leal a alguém

$$\forall x.\exists y.leal(x,y)$$

$$leal(x_3, f(x_3))$$

- Os homens só tentam assassinar soberanos aos quais não são leais

$$\forall x.\forall y.homem(x) \wedge soberano(y) \wedge tentaAssassinar(x,y) \rightarrow \neg leal(x,y)$$

$$\neg homem(x_4) \vee \neg soberano(y_1) \vee \neg tentaAssassinar(x_4, y_1) \vee \neg leal(x_4, y_1)$$

Método da Resolução

- O método da Resolução é um método de prova por *refutação*
  - para provar que $G \models W$, prova-se que $H = G \cup \{\neg W\}$ é *insatisfazível*, isto é, que não existe nenhuma interpretação que satisfaça simultaneamente todas as fórmulas de *H*
- o método baseia-se no cálculo de resolventes

$$\left.\begin{array}{r} p \vee \neg r \\ r \vee q \end{array}\right\} p \vee q$$

As proposições precisam ser *unificáveis*

Algoritmo de Unificação

- Encontrar uma *substituição mais geral* que torne duas fórmulas idênticas
  - Exemplo: As fórmulas atômicas

$$Ama(x, carlos, z) \text{ e } Ama(amelia, y, z)$$

podem ser unificadas pela aplicação da substituição

$$\{x/amelia, y/carlos\}$$

resultando na fórmula

$$Ama(amelia, carlos, z)$$

Algoritmo de Unificação

Unify(ϕ_1, ϕ_2)

se ϕ_1 ou ϕ_2 for um símbolo atômico **então**

se $\phi_1 = \phi_2$ **então retorne** $\{\}$

se não

se ϕ_1 for variável e $\phi_1 \not\rightsquigarrow \phi_2$ **então retorne** $\{\phi_1/\phi_2\}$

se não

se ϕ_2 for variável e $\phi_2 \not\rightsquigarrow \phi_1$ **então retorne** $\{\phi_2/\phi_1\}$

se não retorne *Falha*

se não

Algoritmo de Unificação

Seja $\phi_1 = \varphi_1(t_{11}, \ldots, t_{1n_1})$ e

$\phi_2 = \varphi_2(t_{21}, \ldots, t_{2n_2})$, onde $\varphi_i \in \mathbf{P}$ ou $\varphi_i \in \mathbf{F}$

se $\varphi_1 = \varphi_2$ e $n_1 = n_2 = n$ **então**

$\Theta \leftarrow \emptyset$

para $i = 1, \ldots, n$ **faça**

$\theta \leftarrow$ *Unify*(t_{1i}, t_{2i})

se $\theta =$ *Falha* **então retorne** *Falha*

se não

para $j = i + 1, \ldots, n$ **faça**

$t_{1j} \leftarrow t_{1j}\theta$

$t_{2j} \leftarrow t_{2j}\theta$

$\Theta \leftarrow \Theta \cup \theta$

retorne Θ

se não retorne *Falha*

Algoritmo de Unificação

- O algoritmo consiste em dois casos principais
  - Caso 1: quando uma das fórmulas é um símbolo atômico (variável ou símbolo de constante)
    - as duas fórmulas são símbolos e estes são iguais, resultando na substituição vazia
    - uma das fórmulas é uma variável que não ocorre na outra, resultando em uma substituição com um único par
    - outros casos, resultando em falha

Algoritmo de Unificação

- Caso 2: quando ambas as fórmulas são fórmulas atômicas ou termos compostos por mais de um símbolo
  - se ambas as fórmulas forem termos de mesma aridade e mesmo símbolo de predicado ou função, então o resultado é a composição das diversas substituições necessárias para unificar os subtermos correspondentes em ambas as fórmulas
  - caso algum par de subtermos não seja unificável, ou caso não seja possível combinar as diversas substituições, o algoritmo retorna falha

Algoritmo de Unificação

- Exemplos:

  A = A ana = ana f(A) = f(B) f(g(A), A) = f(B, xyz)

- Contra-exemplos:

  f(A) = g(B) ana = paula A = f(A)

Regra de resolução

- Considere o seguinte par de cláusulas:

$$C_1 = L_{1,1} \vee \cdots \vee L_{1,n}$$
$$C_2 = L_{2,1} \vee \cdots \vee L_{2,m}$$

onde existem *i* e *j* tal que $L_{1,i} = P(t_1, \ldots, t_k)$ e $L_{2,j} = \neg P(t'_1, \ldots, t'_k)$ e existe uma substituição θ, tal que:

$$P(t_1, \ldots, t_k)\theta = P(t'_1, \ldots, t'_k)\theta.$$

- Neste caso, é possível inferir, a partir de C_1 e C_2 e utilizando a regra de resolução, a seguinte cláusula, chamada de *resolvente*:

$$(C_1 - \{L_{1,i}\} \cup C_2 - \{L_{2,j}\})\theta.$$

Algoritmo de resolução

1. Converta G para a forma clausal
2. Negue W e converta para a forma clausal. Acrescente-o ao conjunto de cláusulas obtidas na etapa 1.
3. Repita até que uma contradição seja encontrada ou até que nenhum progresso a mais possa ser feito:
   (a) Selecione duas cláusulas. Chame-as de cláusulas-pai.
   (b) Resolva as duas juntas. A cláusula resultante, chamada de *resolvente*, será a disjunção de todos os literais de ambas as cláusulas-pai com a seguinte excessão: se houver pares de literais L e $\neg L$ tais que uma das cláusulas-pai contenha L e a outra $\neg L$, selecione um desses pares e elimine tanto L quanto $\neg L$ do resolvente.
   (c) Se o resolvente for uma cláusula vazia, foi encontrada uma contradição. Se não for, acresecente-o ao conjunto de cláusulas disponíveis ao procedimento.

Algoritmo de resolução

- As provas por resolução são normalmente apresentadas na forma de uma árvore binária invertida com a cláusula vazia na raiz e as cláusulas do conjunto original nas folhas
- Cada nodo interno é associado a um resolvente e tem como descendentes nodos associados às cláusulas que o geraram

Método da resolução

- Exemplo 1: Considere as seguintes fórmulas lógicas

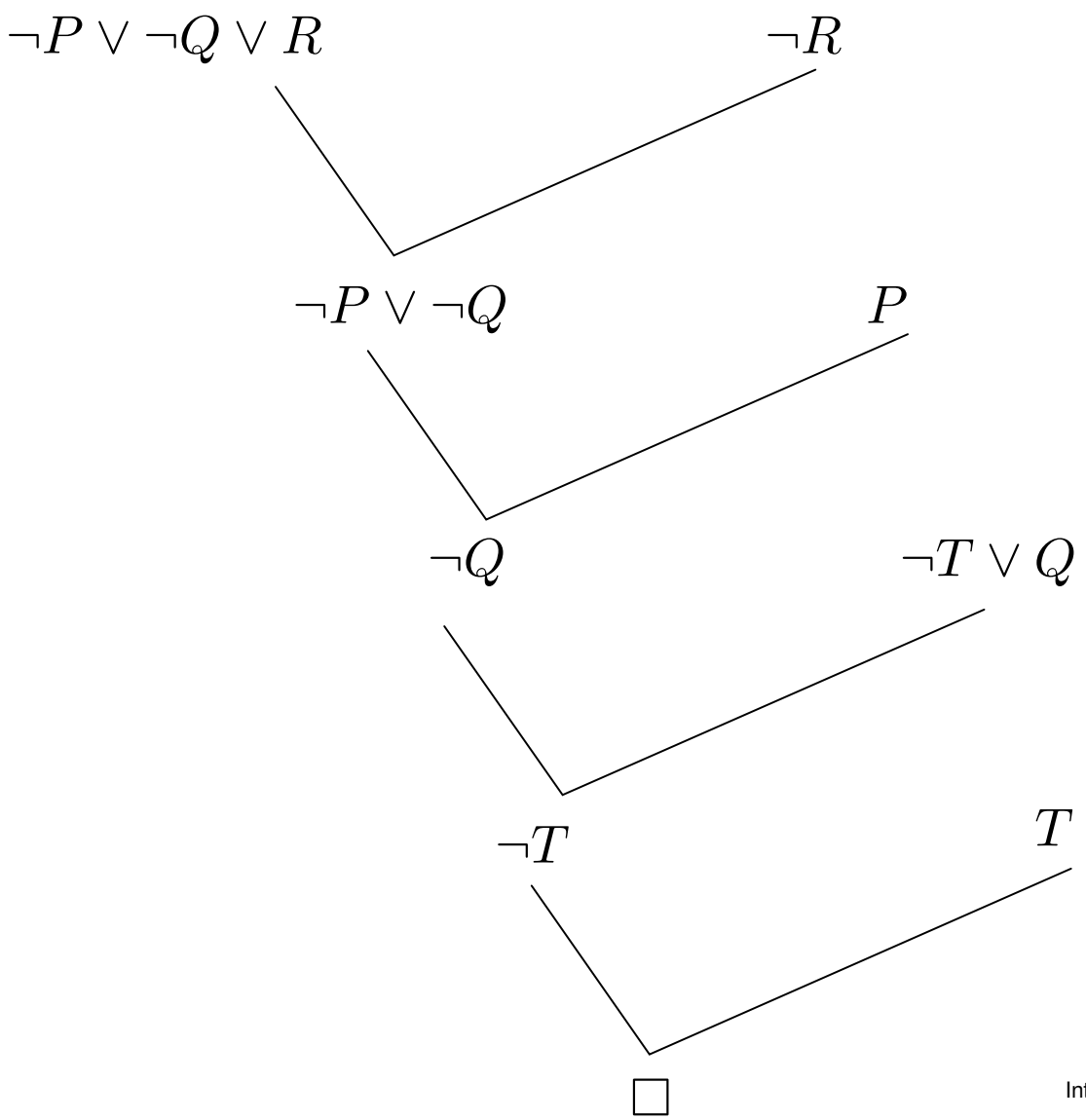
Axiomas	Forma Clausal
P	P
$(P \wedge Q) \rightarrow R$	$\neg P \vee \neg Q \vee R$
$(S \vee T) \rightarrow Q$	$\neg S \vee Q$
	$\neg T \vee Q$
T	T

$W = R$

$H = G \cup \{\neg W\}$

Método da resolução

- Exemplo 2:



Método da resolução

- Exemplo 2: Considere as seguintes fórmulas lógicas

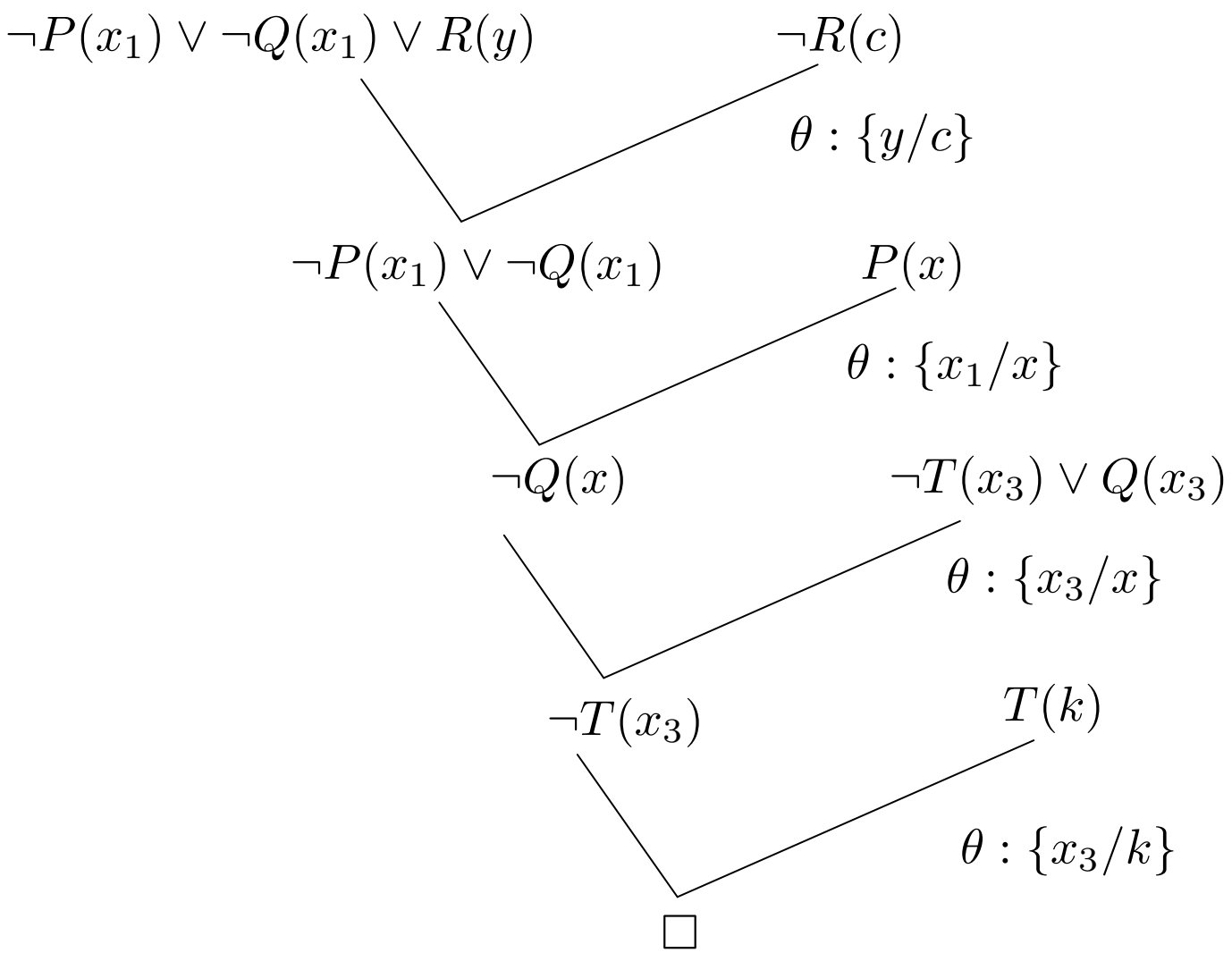
Axiomas	Forma Clausal
$\forall x.P(x)$	$P(x)$
$\forall x.\forall y.P(x) \wedge Q(x) \rightarrow R(y)$	$\neg P(x_1) \vee \neg Q(x_1) \vee R(y)$
$\forall x.S(x) \vee T(x) \rightarrow Q(x)$	$\neg S(x_2) \vee Q(x_2)$
	$\neg T(x_3) \vee Q(x_3)$
$\exists k.T(k)$	$T(k)$

$W = R(c)$

$H = G \cup \{\neg W\}$

Método da resolução

- Exemplo 2:



Método da resolução

- Exemplo 3:

$$W = \quad odeia(marcos, cesar)$$

$$\begin{aligned} G = \quad & \{1.homem(marcos), \\ & 2.pompeano(marcos), \\ & 3.\neg pompeano(x_1) \vee romano(x_1), \\ & 4.soberano(cesar), \\ & 5.\neg romano(x_2) \vee leal(x_2, cesar) \vee odeia(x_2, cesar), \\ & 6.leal(x_3, f(x_3)), \\ & 7.\neg homem(x_4) \vee \neg soberano(y_1) \vee \neg tentaAssassinar(x_4, y_1) \vee \\ & \neg leal(x_4, y_1), \\ & 8.tentaAssassinar(marcos, cesar)\} \end{aligned}$$

Método de Tableaux

- Análogo ao método da Resolução, Tableaux é um método de prova por refutação
- Seja H um conjunto de fórmula em Forma Normal Disjuntiva, se a fórmula é insatisfazível, cada cláusula dual deve ser independentemente insatisfazível (definição do operador $\vee$)
  - cada D_i deve conter uma contradição na forma de um par de literais com sinais inversos e cujas fórmulas atômicas são unificáveis
  - não é necessário completar a transformação para a forma normal, pois se durante a construção de uma cláusula dual for encontrada uma contradição, esta cláusula dual pode ser abandonada

Método de Tableaux

- Regras para a construção de um tableau lógico:
  - Regras para fórmulas conjuntivas:

$$\frac{A \wedge B}{\begin{array}{c} A \\ B \end{array}} \qquad \frac{\neg(A \vee B)}{\begin{array}{c} \neg A \\ \neg B \end{array}} \qquad \frac{\neg(A \rightarrow B)}{\begin{array}{c} A \\ \neg B \end{array}}$$

  - Regras para fórmulas disjuntivas:

$$\begin{array}{c|c} A \vee B & \\ \hline A & B \end{array} \qquad \begin{array}{c|c} \neg(A \wedge B) & \\ \hline \neg A & \neg B \end{array} \qquad \begin{array}{c|c} A \rightarrow B & \\ \hline \neg A & B \end{array}$$

Método de Tableaux

- Regras para a construção de um tableau lógico:
  - Regra para a negação:

$$\frac{\neg\neg A}{A}$$

  - Regras para fórmulas quantificadas universalmente:

$$\frac{\forall x.A}{A\{x/t\}} \qquad \frac{\neg\exists x.A}{\neg A\{x/t\}}$$

onde t é um termo.

  - Regras para fórmulas quantificadas existencialmente:

$$\frac{\exists x.A}{A\{x/\pi\}} \qquad \frac{\neg\forall x.A}{\neg A\{x/\pi\}}$$

onde π é um parâmetro.

Formalmente, é necessário definir uma nova linguagem $L^{\Pi}(\mathbf{P}, \mathbf{F}, \mathbf{C} \cup \Pi, \mathbf{V})$ onde Π é um novo conjunto de constantes, disjunto de **C**, chamado *parâmetros*.

Método de Tableaux

- Formalmente, um tableau lógico para um conjunto de fórmulas $H = \{H_1, \ldots, H_n\}$ é definido da seguinte maneira:

  1. A árvore abaixo, que contém apenas um ramo, é um tableau de *H*.

  A numeração das linhas visa apenas facilitar a referência às fórmulas do tableau.

(1)	H_1
$\vdots$	$\vdots$
(n)	H_n

Método de Tableaux

2. Se um ramo de um tableau de H contém uma fórmula conjuntiva, então o tableau resultante da extensão deste ramo com as duas subfórmulas que formam a fórmula conjuntiva, de acordo com as regras para fórmulas conjuntivas, é um tableau para H.

(1)	H_1		
$\vdots$	$\vdots$		
(i)	$A \wedge B$		
$\vdots$	$\vdots$		
(n)	H_n		
$(n+1)$	A	de	(i)
$(n+2)$	B	de	(i)

Método de Tableaux

3. Se um ramo de um tableau de H contém uma fórmula disjuntiva, então o tableau resultante da bifurcação deste ramo em dois ramos contendo cada um uma das duas subfórmulas que formam a fórmula disjuntiva, de acordo com as regras para fórmulas disjuntivas, é um tableau para H.

(1)	H_1	
$\vdots$	$\vdots$	
(i)	$A \vee B$	
$\vdots$	$\vdots$	
(n)	H_n	
$(n+1)$	A de (i)	B de (i)

Método de Tableaux

4. Se um ramo de um tableau de H contém uma fórmula duplamente negada, então o tableau resultante da extensão deste ramo com a fórmula não negada, de acordo com a regra para a negação, é um tableau para H.

(1)	H_1	
$\vdots$	$\vdots$	
(i)	$\neg\neg A$	
$\vdots$	$\vdots$	
(n)	H_n	
$(n+1)$	A	de (i)

Método de Tableaux

5. Se um ramo de um tableau de H contém uma fórmula quantificada, então o tableau resultante da extensão deste ramo com uma instância da fórmula quantificada, de acordo com as regras para fórmulas quantificadas, é um tableau para H.

(1)	H_1	
$\vdots$	$\vdots$	
(i)	$\forall x.A$	
$\vdots$	$\vdots$	
(n)	H_n	
$(n+1)$	$A\{x/t\}$	de (i)

Método de Tableaux

- Um ramo de um tableu é dito **fechado** se ele contém uma contradição, isto é, duas fórmulas com sinais contrários, x e $\neg x$, onde x é uma fórmula qualquer
- dicas:
  - Negar o teorema
  - Sempre que houver escolha, utilizar antes as regras conjuntivas ou de negação e depois as regras disjuntivas
  - Numerar cada fórmula da prova e indicar, se for o caso, de qual outra fórmula ela foi gerada e o tipo de regra utilizada (conjuntiva, disjuntiva ou negação)
  - Expandir sempre as fórmulas quantificadas existencialmente antes das quantificadas universalmente

Método de Tableaux

- Exemplo:

$$W \quad = \quad \forall x.(Bom(x) \rightarrow Alegria) \rightarrow$$
$$\exists x.Bom(x) \rightarrow Alegria$$

(1)	$\neg(\forall x.(B(x) \rightarrow A) \rightarrow \exists x.B(x) \rightarrow A)$				
(2)	$\forall x.(B(x) \rightarrow A)$			de	(1)
(3)	$\neg(\exists x.B(x) \rightarrow A)$			de	(1)
(4)	$\exists x.B(x)$			de	(3)
(5)	$\neg A$			de	(3)
(6)	$B(a)$			de	(4)
(7)	$B(a) \rightarrow A$			de	(2)
(8)	$\neg B(a)$	de (7)	A	de	(7)

Método de Tableaux

- Caso se queira provar que uma fórmula W é uma conseqüência lógica de um conjunto de fórmulas S, isto é, $S \models W$, é necessário adotar a *regra de introdução de hipóteses*

  Qualquer $x \in S$ pode ser adicionado a qualquer ramo do tableau.

- Diz-se que W pode ser provada a partir de S pelo método de tableaux, notado $S \vdash W$, se, permitindo a introdução de hipóteses existe um tableau fechado para $\neg W$.

Método de Tableaux

- Exemplo:

$$\{\forall x.(Homem(x) \rightarrow Mortal(x)), Homem(socrates)\} \vdash Mortal(socrates)$$

(1)	$\neg M(s)$	$\neg W$
(2)	$\forall x.(H(x) \rightarrow M(x))$	R.I.
(3)	$(H(s) \rightarrow M(s))$	de (2)
(4)	$\neg H(s)$ de (3)	$M(s)$ de (3)
(5)	$H(s)$ R.I.	

Prolog

Histórico

- Acrônimo de *PROgramming in LOGic*
- Desenvolvida na década de 70 por Alain Colmerauer e Phillipe Roussel (Artificial Intelligence Group - Université Aix-Marseille) e Robert Kowalski (Departament of Artificial Intelligence - University of Edimburgh)
- 1972 - Desenvolvimento do 1^o interpretador PROLOG em Marseille
- Meados de 70 - fim da cooperação
  - surgimento de 2 dialetos distintos

Prolog

Histórico

- 1981 - Projeto Fifth Generation Computing Systems (Japão)
  - desenvolvimento de máquinas inteligentes
  - atenção voltada para a Inteligência Artificial e a Programação em Lógica
- Surgimento de variações de PROLOG com diferentes sintaxes

Prolog

- um programa é constituido por dois elementos disjuntos:
  - lógica (descrição dos fatos)
  - controle (mecanismo de solução)
- O programador deve preocupar-se somente em descrever o problema a ser solucionado (especificação), deixando a critério da máquina a busca pela solução

Prolog

- A descrição dos fatos é feita por meio de *cláusulas* (G)
  - fatos
  - regras
  - consultas (W)

Tipos de Cláusulas

- Uma implicação lógica é da forma $A \rightarrow B$
  - onde onde A esta na forma:

$$A_1(x_1, ..., x_k) \wedge ... \wedge A_m(x_1, ..., x_k)$$

  - e B está na forma

$$B_1(x_1, ..., x_k) \vee ... \vee B_n(x_1, ..., x_k)$$

- Usando a relação de equivalência temos $\neg A \vee B$

Tipos de Cláusulas

- Geral:

$$A_1(x_1, ..., x_k) \wedge ... \wedge A_m(x_1, ..., x_k) \rightarrow B_1(x_1, ..., x_k) \vee ... \vee B_n(x_1, ..., x_k)$$

- Positiva:

$$B_1(x_1, ..., x_k) \vee ... \vee B_n(x_1, ..., x_k)$$

- Fechada:

$$A_1 \wedge ... \wedge A_m \rightarrow B_1 \vee ... \vee B_n$$

- Positiva Fechada:

$$B_1 \vee ... \vee B_n$$

Tipos de Cláusulas

- Cláusula de Horn Geral:

$$A_1(x_1, ..., x_k) \wedge ... \wedge A_m(x_1, ..., x_k) \rightarrow B_1(x_1, ..., x_k)$$

- Cláusula de Horn Positiva:

$$B_1(x_1, ..., x_k)$$

- Cláusula de Horn Fechada:

$$A_1 \wedge ... \wedge A_m \rightarrow B_1$$

- Cláusula de Horn Positiva Fechada:

$$B_1$$

Cláusulas em PROLOG

- A linguagem Prolog utiliza apenas cláusulas de Horn
- Também é adotada uma notação diferente
  - Variáveis são declaradas em letras maiúsculas
  - Constantes são declaradas em letras minúsculas

Cláusulas em PROLOG

- os quatro tipos de cláusulas de Horn são escritos em Prolog como:

$$B_1(x_1, ..., x_k) :- A_1(x_1, ..., x_k) \wedge ... \wedge A_m(x_1, ..., x_k).$$

$$B_1(x_1, ..., x_k).$$

$$B_1 :- A_1 \wedge ... \wedge A_m.$$

$$B1.$$

Cláusulas em PROLOG

- cada cláusula em prolog possui uma cabeça (antes de $:-$) e um corpo (depois de $:-$)
  - uma cláusula com corpo vazio é um **fato**
  - caso contrário, a cláusula (**regra**) será verdade se o corpo for verdade

$$a :- b, c, d \Rightarrow a \text{ é verdade se } b \wedge c \wedge d \text{ é verdade}$$

  - $b \wedge c \wedge d \rightarrow a$ ou em cláusula de Horn $\neg b \vee \neg c \vee \neg d \vee a$

Cláusulas em PROLOG

- Exemplo:

homem(marcos). homem(aquiles). homem(brutos).

pompeano(marcos). pompeano(aquiles). pompeano(brutos).

soberano(cesar).

tentaAssassinar(marcos, cesar).

romano(X):- pompeano(X).

leal(X, cesar):- romano(X), $\sim$odeia(X,cesar).

odeia(X,Y):- homem(X), soberano(Y), tentaAssassinar(X,Y).

Referências

- G. Bittencourt, *Inteligência Artificial: Ferramentas e Teorias*, 3^a Edição, Editora da UFSC, Florianópolis, SC, 2006.
- M.A. Casanova, F.A.C. Giorno, A.L. Furtado, *Programação em Lógica e a Linguagem Prolog*, 2006.
- J.M. Barreto, *Inteligência Artificial - No limiar do Século XXI*, 1999.