

# Inteligência Artificial – ACH2016

## Aula 11 – Inferência em Lógica de Primeira Ordem

Norton Trevisan Roman  
(norton@usp.br)

5 de abril de 2019

# Bases em Lógica de Primeira Ordem

## Interagindo com uma BC

- Fazemos buscas (queries) na base

# Bases em Lógica de Primeira Ordem

## Interagindo com uma BC

- Fazemos buscas (queries) na base
  - Feitas por meio de sentenças lógicas em LPO

# Bases em Lógica de Primeira Ordem

## Interagindo com uma BC

- Fazemos buscas (queries) na base
  - Feitas por meio de sentenças lógicas em LPO
  - Qualquer *query* que possa ser inferida logicamente pela base de dados será respondida afirmativamente

# Bases em Lógica de Primeira Ordem

## Interagindo com uma BC

- Fazemos buscas (queries) na base
  - Feitas por meio de sentenças lógicas em LPO
  - Qualquer *query* que possa ser inferida logicamente pela base de dados será respondida afirmativamente
- A BC é composta por:

# Bases em Lógica de Primeira Ordem

## Interagindo com uma BC

- Fazemos buscas (queries) na base
  - Feitas por meio de sentenças lógicas em LPO
  - Qualquer *query* que possa ser inferida logicamente pela base de dados será respondida afirmativamente
- A BC é composta por:
  - **Axiomas:** informação factual básica da qual conclusões úteis podem ser derivadas

# Bases em Lógica de Primeira Ordem

## Interagindo com uma BC

- Fazemos buscas (queries) na base
  - Feitas por meio de sentenças lógicas em LPO
  - Qualquer *query* que possa ser inferida logicamente pela base de dados será respondida afirmativamente
- A BC é composta por:
  - **Axiomas:** informação factual básica da qual conclusões úteis podem ser derivadas
    - Ex:  $\forall p, f \text{ Progenitor}(p, f) \Leftrightarrow \text{Prole}(f, p); \text{Homem}(\text{Pedro}); \text{etc}$

# Bases em Lógica de Primeira Ordem

## Interagindo com uma BC

- Fazemos buscas (queries) na base
  - Feitas por meio de sentenças lógicas em LPO
  - Qualquer *query* que possa ser inferida logicamente pela base de dados será respondida afirmativamente
- A BC é composta por:
  - **Axiomas:** informação factual básica da qual conclusões úteis podem ser derivadas
    - Ex:  $\forall p, f \text{ Progenitor}(p, f) \Leftrightarrow \text{Prole}(f, p); \text{Homem}(\text{Pedro}); \text{etc}$
  - **Teoremas:** sentenças acarretadas pelos axiomas



# Bases em Lógica de Primeira Ordem

## Interagindo com uma BC

- Fazemos buscas (queries) na base
  - Feitas por meio de sentenças lógicas em LPO
  - Qualquer *query* que possa ser inferida logicamente pela base de dados será respondida afirmativamente
- A BC é composta por:
  - **Axiomas:** informação factual básica da qual conclusões úteis podem ser derivadas
    - Ex:  $\forall p, f \text{ Progenitor}(p, f) \Leftrightarrow \text{Prole}(f, p); \text{Homem}(\text{Pedro});$  etc
  - **Teoremas:** sentenças acarretadas pelos axiomas
    - Não precisam existir na base, bastam os axiomas

# Bases em Lógica de Primeira Ordem

## Interagindo com uma BC

- Fazemos buscas (queries) na base
  - Feitas por meio de sentenças lógicas em LPO
  - Qualquer *query* que possa ser inferida logicamente pela base de dados será respondida afirmativamente
- A BC é composta por:
  - **Axiomas:** informação factual básica da qual conclusões úteis podem ser derivadas
    - Ex:  $\forall p, f \text{ Progenitor}(p, f) \Leftrightarrow \text{Prole}(f, p); \text{Homem}(\text{Pedro});$  etc
  - **Teoremas:** sentenças acarretadas pelos axiomas
    - Não precisam existir na base, bastam os axiomas
    - Reduzem o custo computacional de derivar novas sentenças

# Lógica de Primeira Ordem

## Inferência

# Lógica de Primeira Ordem

## Inferência

- Como em lógica proposicional

## Inferência

- Como em lógica proposicional
  - Uma sentença pode ser inferida de uma BC sse for verdadeira em toda interpretação na qual a BC é verdadeira

## Inferência

- Como em lógica proposicional
  - Uma sentença pode ser inferida de uma BC sse for verdadeira em toda interpretação na qual a BC é verdadeira
- Força bruta está fora de questão

## Inferência

- Como em lógica proposicional
  - Uma sentença pode ser inferida de uma BC sse for verdadeira em toda interpretação na qual a BC é verdadeira
- Força bruta está fora de questão
  - Número enorme (possivelmente  $\infty$ ) de interpretações

# Lógica de Primeira Ordem

## Inferência

- Como em lógica proposicional
  - Uma sentença pode ser inferida de uma BC sse for verdadeira em toda interpretação na qual a BC é verdadeira
- Força bruta está fora de questão
  - Número enorme (possivelmente  $\infty$ ) de interpretações
  - Quantificadores  $\forall$  e  $\exists$



# Lógica de Primeira Ordem

## Inferência

- Como em lógica proposicional
  - Uma sentença pode ser inferida de uma BC sse for verdadeira em toda interpretação na qual a BC é verdadeira
- Força bruta está fora de questão
  - Número enorme (possivelmente  $\infty$ ) de interpretações
  - Quantificadores  $\forall$  e  $\exists$ 
    - Para  $\exists$  existe skolemização (mais adiante), contudo...

# Lógica de Primeira Ordem

## Inferência

- Como em lógica proposicional
  - Uma sentença pode ser inferida de uma BC sse for verdadeira em toda interpretação na qual a BC é verdadeira
- Força bruta está fora de questão
  - Número enorme (possivelmente  $\infty$ ) de interpretações
  - Quantificadores  $\forall$  e  $\exists$ 
    - Para  $\exists$  existe skolemização (mais adiante), contudo...
    - $\forall$  pode mapear a infinitos elementos, dependendo do domínio

# Lógica de Primeira Ordem

## Inferência

- Como em lógica proposicional
  - Uma sentença pode ser inferida de uma BC sse for verdadeira em toda interpretação na qual a BC é verdadeira
- Força bruta está fora de questão
  - Número enorme (possivelmente  $\infty$ ) de interpretações
  - Quantificadores  $\forall$  e  $\exists$ 
    - Para  $\exists$  existe skolemização (mais adiante), contudo...
    - $\forall$  pode mapear a infinitos elementos, dependendo do domínio
- Solução: provas

# Lógica de Primeira Ordem – Inferência

## Provas

- Há versões de primeira ordem para cada técnica vista em lógica proposicional

# Lógica de Primeira Ordem – Inferência

## Provas

- Há versões de primeira ordem para cada técnica vista em lógica proposicional
  - Forward Chaining, Backward Chaining e Resolução

# Lógica de Primeira Ordem – Inferência

## Provas

- Há versões de primeira ordem para cada técnica vista em lógica proposicional
  - Forward Chaining, Backward Chaining e Resolução
- Forward Chaining e Backward Chaining exigem que a base contenha apenas cláusulas definidas

# Lógica de Primeira Ordem – Inferência

## Provas

- Há versões de primeira ordem para cada técnica vista em lógica proposicional
  - Forward Chaining, Backward Chaining e Resolução
- Forward Chaining e Backward Chaining exigem que a base contenha apenas cláusulas definidas
  - Disjunções de literais dos quais exatamente um é positivo

# Lógica de Primeira Ordem – Inferência

## Provas

- Há versões de primeira ordem para cada técnica vista em lógica proposicional
  - Forward Chaining, Backward Chaining e Resolução
- Forward Chaining e Backward Chaining exigem que a base contenha apenas cláusulas definidas
  - Disjunções de literais dos quais exatamente um é positivo
- Literais



# Lógica de Primeira Ordem – Inferência

## Provas

- Há versões de primeira ordem para cada técnica vista em lógica proposicional
  - Forward Chaining, Backward Chaining e Resolução
- Forward Chaining e Backward Chaining exigem que a base contenha apenas cláusulas definidas
  - Disjunções de literais dos quais exatamente um é positivo
- Literais
  - Literais em LPO podem incluir variáveis

# Lógica de Primeira Ordem – Inferência

## Provas

- Há versões de primeira ordem para cada técnica vista em lógica proposicional
  - Forward Chaining, Backward Chaining e Resolução
- Forward Chaining e Backward Chaining exigem que a base contenha apenas cláusulas definidas
  - Disjunções de literais dos quais exatamente um é positivo
- Literais
  - Literais em LPO podem incluir variáveis
    - “*Amigo(Pedro)*” (constante)

# Lógica de Primeira Ordem – Inferência

## Provas

- Há versões de primeira ordem para cada técnica vista em lógica proposicional
  - Forward Chaining, Backward Chaining e Resolução
- Forward Chaining e Backward Chaining exigem que a base contenha apenas cláusulas definidas
  - Disjunções de literais dos quais exatamente um é positivo
- Literais
  - Literais em LPO podem incluir variáveis
    - “*Amigo(Pedro)*” (constante)
    - “*Grande(x)*” (variável)

## Provas

- Em qualquer técnica, expressões com variáveis são assumidas como universalmente quantificadas

## Provas

- Em qualquer técnica, expressões com variáveis são assumidas como universalmente quantificadas
  - Ou seja, todas são naturalmente precedidas por  $\forall$

## Provas

- Em qualquer técnica, expressões com variáveis são assumidas como universalmente quantificadas
  - Ou seja, todas são naturalmente precedidas por  $\forall$
  - “*Grande(x)*”  $\equiv$  “ $\forall x$  *Grande(x)*” (literal)

## Provas

- Em qualquer técnica, expressões com variáveis são assumidas como universalmente quantificadas
  - Ou seja, todas são naturalmente precedidas por  $\forall$
  - “*Grande(x)*”  $\equiv$  “ $\forall x$  *Grande(x)*” (literal)
  - “*Tio(x, y)  $\Leftrightarrow$  Sobrinho(y, x)*”  $\equiv$   
“ $\forall x, y$  *Tio(x, y)  $\Leftrightarrow$  Sobrinho(y, x)*”

# Lógica de Primeira Ordem – Inferência

## Provas

- Em qualquer técnica, expressões com variáveis são assumidas como universalmente quantificadas
  - Ou seja, todas são naturalmente precedidas por  $\forall$
  - “*Grande*( $x$ )”  $\equiv$  “ $\forall x$  *Grande*( $x$ )” (literal)
  - “*Tio*( $x, y$ )  $\Leftrightarrow$  *Sobrinho*( $y, x$ )”  $\equiv$   
“ $\forall x, y$  *Tio*( $x, y$ )  $\Leftrightarrow$  *Sobrinho*( $y, x$ )”
- Começaremos pela resolução, por ser mais geral



## Resolução

# Lógica de Primeira Ordem – Resolução

## Passos

# Lógica de Primeira Ordem – Resolução

## Passos

- 1 Converter de LPO para forma clausal

$$\forall x \exists y P(x) \Rightarrow R(x, y)$$



$$\neg P(x) \vee R(x, F(x))$$

# Lógica de Primeira Ordem – Resolução

## Passos

### 1 Converter de LPO para forma clausal

- Generalização da Forma Normal Conjuntiva para LPO

$$\forall x \exists y P(x) \Rightarrow R(x, y)$$



$$\neg P(x) \vee R(x, F(x))$$

# Lógica de Primeira Ordem – Resolução

## Passos

### 1 Converter de LPO para forma clausal

- Generalização da Forma Normal Conjuntiva para LPO
- Uma conjunção de disjunções

$$\forall x \exists y P(x) \Rightarrow R(x, y)$$



$$\neg P(x) \vee R(x, F(x))$$

# Lógica de Primeira Ordem – Resolução

## Passos

### 1 Converter de LPO para forma clausal

- Generalização da Forma Normal Conjuntiva para LPO
- Uma conjunção de disjunções
- Sem quantificadores

$$\forall x \exists y P(x) \Rightarrow R(x, y)$$



$$\neg P(x) \vee R(x, F(x))$$

# Lógica de Primeira Ordem – Resolução

## Passos

### 1 Converter de LPO para forma clausal

- Generalização da Forma Normal Conjuntiva para LPO
- Uma conjunção de disjunções
- Sem quantificadores

$$\forall x \exists y P(x) \Rightarrow R(x, y)$$



$$\neg P(x) \vee R(x, F(x))$$

### 2 Determinar que variáveis substituir por quais quando da resolução

# Lógica de Primeira Ordem – Resolução

## Passos

### 1 Converter de LPO para forma clausal

- Generalização da Forma Normal Conjuntiva para LPO
- Uma conjunção de disjunções
- Sem quantificadores

$$\forall x \exists y P(x) \Rightarrow R(x, y)$$



$$\neg P(x) \vee R(x, F(x))$$

- ### 2 Determinar que variáveis substituir por quais quando da resolução
- Processo chamado Unificação



# Lógica de Primeira Ordem – Resolução

## Passos

### 1 Converter de LPO para forma clausal

- Generalização da Forma Normal Conjuntiva para LPO
- Uma conjunção de disjunções
- Sem quantificadores

$$\forall x \exists y P(x) \Rightarrow R(x, y)$$



$$\neg P(x) \vee R(x, F(x))$$

### 2 Determinar que variáveis substituir por quais quando da resolução

- Processo chamado Unificação

### 3 Executar a resolução

# Lógica de Primeira Ordem – Resolução

## Passos

### 1 Converter de LPO para forma clausal 📌

- Generalização da Forma Normal Conjuntiva para LPO
- Uma conjunção de disjunções
- Sem quantificadores

$$\forall x \exists y P(x) \Rightarrow R(x, y)$$



$$\neg P(x) \vee R(x, F(x))$$

- ### 2 Determinar que variáveis substituir por quais quando da resolução
- Processo chamado Unificação
- ### 3 Executar a resolução

## Forma Normal Conjuntiva

- Como no caso proposicional, resolução em LPO também exige que as sentenças estejam na FNC

# Lógica de Primeira Ordem – Resolução

## Forma Normal Conjuntiva

- Como no caso proposicional, resolução em LPO também exige que as sentenças estejam na FNC
- Conjunção de cláusulas, onde cada cláusula é uma disjunção de literais

## Forma Normal Conjuntiva

- Como no caso proposicional, resolução em LPO também exige que as sentenças estejam na FNC
- Conjunção de cláusulas, onde cada cláusula é uma disjunção de literais
- Lembrando que literais podem conter variáveis universalmente quantificadas

## Forma Normal Conjuntiva

- Como no caso proposicional, resolução em LPO também exige que as sentenças estejam na FNC
- Conjunção de cláusulas, onde cada cláusula é uma disjunção de literais
- Lembrando que literais podem conter variáveis universalmente quantificadas
- Também chamada de **Forma Clausal**

## Forma Normal Conjuntiva

- Como no caso proposicional, resolução em LPO também exige que as sentenças estejam na FNC
- Conjunção de cláusulas, onde cada cláusula é uma disjunção de literais
- Lembrando que literais podem conter variáveis universalmente quantificadas
- Também chamada de **Forma Clausal**
- Também como no caso proposicional, toda sentença em LPO pode ser convertida em uma sentença inferencialmente equivalente na FNC

## Equivalência Inferencial

- Duas sentenças são tidas como **inferencialmente equivalentes** quando uma for satisfatível sse a outra for satisfatível também



## Equivalência Inferencial

- Duas sentenças são tidas como **inferencialmente equivalentes** quando uma for satisfatível sse a outra for satisfatível também
- Alternativamente, uma será insatisfatível somente quando a outra também o for

## Equivalência Inferencial

- Duas sentenças são tidas como **inferencialmente equivalentes** quando uma for satisfatível sse a outra for satisfatível também
- Alternativamente, uma será insatisfatível somente quando a outra também o for
- $A \vdash B$  e  $B \vdash A$

## Equivalência Inferencial

- Duas sentenças são tidas como **inferencialmente equivalentes** quando uma for satisfatível sse a outra for satisfatível também
- Alternativamente, uma será insatisfatível somente quando a outra também o for
- $A \vdash B$  e  $B \vdash A$
- Elas não precisam ser logicamente equivalentes

## Equivalência Inferencial

- Duas sentenças são tidas como **inferencialmente equivalentes** quando uma for satisfatível sse a outra for satisfatível também
  - Alternativamente, uma será insatisfatível somente quando a outra também o for
  - $A \vdash B$  e  $B \vdash A$
- Elas não precisam ser logicamente equivalentes
  - Apenas permitem que o mesmo conjunto de inferências seja feito

# Lógica de Primeira Ordem – Resolução

## Convertendo para Forma Clausal

# Lógica de Primeira Ordem – Resolução

## Convertendo para Forma Clausal

- 1 Elimine implicações

# Lógica de Primeira Ordem – Resolução

## Convertendo para Forma Clausal

- 1 Elimine implicações

$$\begin{aligned}\alpha \Leftrightarrow \beta &\equiv (\alpha \Rightarrow \beta) \wedge (\beta \Rightarrow \alpha) \\ \alpha \Rightarrow \beta &\equiv \neg \alpha \vee \beta\end{aligned}$$

# Lógica de Primeira Ordem – Resolução

## Convertendo para Forma Clausal

1 Elimine implicações

$$\alpha \Leftrightarrow \beta \equiv (\alpha \Rightarrow \beta) \wedge (\beta \Rightarrow \alpha)$$

$$\alpha \Rightarrow \beta \equiv \neg \alpha \vee \beta$$

2 Mova a negação “para dentro”



# Lógica de Primeira Ordem – Resolução

## Convertendo para Forma Clausal

1 Elimine implicações

$$\alpha \Leftrightarrow \beta \equiv (\alpha \Rightarrow \beta) \wedge (\beta \Rightarrow \alpha)$$

$$\alpha \Rightarrow \beta \equiv \neg \alpha \vee \beta$$

2 Mova a negação “para dentro”

- Leis de de Morgan

# Lógica de Primeira Ordem – Resolução

## Convertendo para Forma Clausal

1 Elimine implicações

$$\alpha \Leftrightarrow \beta \equiv (\alpha \Rightarrow \beta) \wedge (\beta \Rightarrow \alpha)$$
$$\alpha \Rightarrow \beta \equiv \neg \alpha \vee \beta$$

2 Mova a negação “para dentro”

- Leis de de Morgan

$$\neg(\alpha \vee \beta) \equiv \neg \alpha \wedge \neg \beta$$
$$\neg(\alpha \wedge \beta) \equiv \neg \alpha \vee \neg \beta$$

# Lógica de Primeira Ordem – Resolução

## Convertendo para Forma Clausal

1 Elimine implicações

$$\alpha \Leftrightarrow \beta \equiv (\alpha \Rightarrow \beta) \wedge (\beta \Rightarrow \alpha)$$
$$\alpha \Rightarrow \beta \equiv \neg\alpha \vee \beta$$

2 Mova a negação “para dentro”

- Leis de de Morgan
- Eliminação de negações duplas

$$\neg(\alpha \vee \beta) \equiv \neg\alpha \wedge \neg\beta$$
$$\neg(\alpha \wedge \beta) \equiv \neg\alpha \vee \neg\beta$$

# Lógica de Primeira Ordem – Resolução

## Convertendo para Forma Clausal

1 Elimine implicações

$$\alpha \Leftrightarrow \beta \equiv (\alpha \Rightarrow \beta) \wedge (\beta \Rightarrow \alpha)$$
$$\alpha \Rightarrow \beta \equiv \neg \alpha \vee \beta$$

2 Mova a negação “para dentro”

- Leis de de Morgan

$$\neg(\alpha \vee \beta) \equiv \neg \alpha \wedge \neg \beta$$

- Eliminação de negações duplas

$$\neg(\alpha \wedge \beta) \equiv \neg \alpha \vee \neg \beta$$

$$\neg \neg \alpha \equiv \alpha$$

# Lógica de Primeira Ordem – Resolução

## Convertendo para Forma Clausal

1 Elimine implicações

$$\alpha \Leftrightarrow \beta \equiv (\alpha \Rightarrow \beta) \wedge (\beta \Rightarrow \alpha)$$
$$\alpha \Rightarrow \beta \equiv \neg \alpha \vee \beta$$

2 Mova a negação “para dentro”

- Leis de de Morgan
- Eliminação de negações duplas
- Extensões a de Morgan

$$\neg(\alpha \vee \beta) \equiv \neg \alpha \wedge \neg \beta$$
$$\neg(\alpha \wedge \beta) \equiv \neg \alpha \vee \neg \beta$$
$$\neg \neg \alpha \equiv \alpha$$

# Lógica de Primeira Ordem – Resolução

## Convertendo para Forma Clausal

1 Elimine implicações

$$\alpha \Leftrightarrow \beta \equiv (\alpha \Rightarrow \beta) \wedge (\beta \Rightarrow \alpha)$$
$$\alpha \Rightarrow \beta \equiv \neg \alpha \vee \beta$$

2 Mova a negação “para dentro”

- Leis de de Morgan
- Eliminação de negações duplas
- Extensões a de Morgan

$$\neg(\alpha \vee \beta) \equiv \neg \alpha \wedge \neg \beta$$

$$\neg(\alpha \wedge \beta) \equiv \neg \alpha \vee \neg \beta$$

$$\neg \neg \alpha \equiv \alpha$$

$$\neg \forall x \alpha \equiv \exists x \neg \alpha$$

$$\neg \exists x \alpha \equiv \forall x \neg \alpha$$

# Lógica de Primeira Ordem – Resolução

## Convertendo para Forma Clausal

1 Elimine implicações

$$\alpha \Leftrightarrow \beta \equiv (\alpha \Rightarrow \beta) \wedge (\beta \Rightarrow \alpha)$$
$$\alpha \Rightarrow \beta \equiv \neg \alpha \vee \beta$$

2 Mova a negação “para dentro”

- Leis de de Morgan
- Eliminação de negações duplas
- Extensões a de Morgan

$$\neg(\alpha \vee \beta) \equiv \neg \alpha \wedge \neg \beta$$

$$\neg(\alpha \wedge \beta) \equiv \neg \alpha \vee \neg \beta$$

$$\neg \neg \alpha \equiv \alpha$$

$$\neg \forall x \alpha \equiv \exists x \neg \alpha$$

$$\neg \exists x \alpha \equiv \forall x \neg \alpha$$

3 Padronize as variáveis

# Lógica de Primeira Ordem – Resolução

## Convertendo para Forma Clausal

### 1 Elimine implicações

$$\alpha \Leftrightarrow \beta \equiv (\alpha \Rightarrow \beta) \wedge (\beta \Rightarrow \alpha)$$
$$\alpha \Rightarrow \beta \equiv \neg \alpha \vee \beta$$

### 2 Mova a negação “para dentro”

- Leis de de Morgan
- Eliminação de negações duplas
- Extensões a de Morgan

$$\neg(\alpha \vee \beta) \equiv \neg \alpha \wedge \neg \beta$$

$$\neg(\alpha \wedge \beta) \equiv \neg \alpha \vee \neg \beta$$

$$\neg \neg \alpha \equiv \alpha$$

$$\neg \forall x \alpha \equiv \exists x \neg \alpha$$

$$\neg \exists x \alpha \equiv \forall x \neg \alpha$$

### 3 Padronize as variáveis

- Cada quantificador deve agir sobre uma variável diferente



# Lógica de Primeira Ordem – Resolução

## Convertendo para Forma Clausal

### 1 Elimine implicações

$$\alpha \Leftrightarrow \beta \equiv (\alpha \Rightarrow \beta) \wedge (\beta \Rightarrow \alpha)$$
$$\alpha \Rightarrow \beta \equiv \neg \alpha \vee \beta$$

### 2 Mova a negação “para dentro”

- Leis de de Morgan
- Eliminação de negações duplas
- Extensões a de Morgan

$$\neg(\alpha \vee \beta) \equiv \neg \alpha \wedge \neg \beta$$

$$\neg(\alpha \wedge \beta) \equiv \neg \alpha \vee \neg \beta$$

$$\neg \neg \alpha \equiv \alpha$$

$$\neg \forall x \alpha \equiv \exists x \neg \alpha$$

$$\neg \exists x \alpha \equiv \forall x \neg \alpha$$

### 3 Padronize as variáveis

- Cada quantificador deve agir sobre uma variável diferente
- Se a sentença usar um mesmo nome de variável duas vezes em escopos diferentes, mude o nome de uma delas

# Lógica de Primeira Ordem – Resolução

## Convertendo para Forma Clausal

### 4 Skolemize

## Convertendo para Forma Clausal

### 4 Skolemize

- **Skolemização** é o processo de remoção de quantificadores existenciais ( $\exists$ ) por eliminação

## Convertendo para Forma Clausal

### 4 Skolemize

- **Skolemização** é o processo de remoção de quantificadores existenciais ( $\exists$ ) por eliminação
- Deve-se a Thoralf Skolem

## Convertendo para Forma Clausal

- ④ Skolemize
  - **Skolemização** é o processo de remoção de quantificadores existenciais ( $\exists$ ) por eliminação
  - Deve-se a Thoralf Skolem
- ⑤ Remova quantificadores universais

## Convertendo para Forma Clausal

### 4 Skolemize

- **Skolemização** é o processo de remoção de quantificadores existenciais ( $\exists$ ) por eliminação
- Deve-se a Thoralf Skolem

### 5 Remova quantificadores universais

- Já removemos  $\exists$  e não temos mais o problema do escopo  $\rightarrow$  as variáveis restantes estão quantificadas universalmente

## Convertendo para Forma Clausal

### 4 Skolemize

- **Skolemização** é o processo de remoção de quantificadores existenciais ( $\exists$ ) por eliminação
- Deve-se a Thoralf Skolem

### 5 Remova quantificadores universais

- Já removemos  $\exists$  e não temos mais o problema do escopo  $\rightarrow$  as variáveis restantes estão quantificadas universalmente
- Podemos então mover os quantificadores para a esquerda e sumir com eles (FNC já assume  $\forall$  nas variáveis)

# Lógica de Primeira Ordem – Resolução

## Convertendo para Forma Clausal

### 4 Skolemize

- **Skolemização** é o processo de remoção de quantificadores existenciais ( $\exists$ ) por eliminação
- Deve-se a Thoralf Skolem

### 5 Remova quantificadores universais

- Já removemos  $\exists$  e não temos mais o problema do escopo  $\rightarrow$  as variáveis restantes estão quantificadas universalmente
- Podemos então mover os quantificadores para a esquerda e sumir com eles (FNC já assume  $\forall$  nas variáveis)

### 6 Distribua $\vee$ sobre $\wedge$



## Skolemização

- No caso do padrão simples “ $\exists x P(x)$ ”, basta substituir a variável  $x$  por uma constante inédita (não existente na base de conhecimento)

## Skolemização

- No caso do padrão simples “ $\exists x P(x)$ ”, basta substituir a variável  $x$  por uma constante inédita (não existente na base de conhecimento)
- Ex:

## Skolemização

- No caso do padrão simples “ $\exists x P(x)$ ”, basta substituir a variável  $x$  por uma constante inédita (não existente na base de conhecimento)
- Ex:
  - $\exists x \text{ Unicórnio}(x)$

## Skolemização

- No caso do padrão simples “ $\exists x P(x)$ ”, basta substituir a variável  $x$  por uma constante inédita (não existente na base de conhecimento)
- Ex:
  - $\exists x \text{ Unicórnio}(x)$
  - Se existe um unicórnio, vamos chamá-lo de *Fred*

## Skolemização

- No caso do padrão simples “ $\exists x P(x)$ ”, basta substituir a variável  $x$  por uma constante inédita (não existente na base de conhecimento)
- Ex:
  - $\exists x \text{ Unicórnio}(x)$
  - Se existe um unicórnio, vamos chamá-lo de *Fred*
  - Funciona se não houver nenhum outro objeto chamado *Fred*, pois nesse caso afirmaríamos que ele é um unicórnio

## Skolemização

- No caso do padrão simples " $\exists x P(x)$ ", basta substituir a variável  $x$  por uma constante inédita (não existente na base de conhecimento)
- Ex:
  - $\exists x \text{ Unicórnio}(x)$
  - Se existe um unicórnio, vamos chamá-lo de *Fred*
  - Funciona se não houver nenhum outro objeto chamado *Fred*, pois nesse caso afirmaríamos que ele é um unicórnio
  - Podemos então trocar  $\exists x \text{ Unicórnio}(x)$  por  $\text{Unicórnio}(\text{Fred})$  livremente

## Skolemização

- Então

## Skolemização

- Então
  - $\exists x P(x) \rightarrow P(C_1)$



## Skolemização

- Então
  - $\exists x P(x) \rightarrow P(C_1)$
  - $\exists x, y R(x, y) \rightarrow R(C_1, C_2)$  (2 variáveis)

## Skolemização

- Então
  - $\exists x P(x) \rightarrow P(C_1)$
  - $\exists x, y R(x, y) \rightarrow R(C_1, C_2)$  (2 variáveis)
  - $\exists x P(x) \wedge Q(x) \rightarrow P(C_1) \wedge Q(C_1)$

## Skolemização

- Então
  - $\exists x P(x) \rightarrow P(C_1)$
  - $\exists x, y R(x, y) \rightarrow R(C_1, C_2)$  (2 variáveis)
  - $\exists x P(x) \wedge Q(x) \rightarrow P(C_1) \wedge Q(C_1)$ 
    - Cada ocorrência da variável é mapeada ao mesmo nome

## Skolemização

- Então
  - $\exists x P(x) \rightarrow P(C_1)$
  - $\exists x, y R(x, y) \rightarrow R(C_1, C_2)$  (2 variáveis)
  - $\exists x P(x) \wedge Q(x) \rightarrow P(C_1) \wedge Q(C_1)$ 
    - Cada ocorrência da variável é mapeada ao mesmo nome
  - $\exists x P(x) \wedge \exists x Q(x) \rightarrow P(C_1) \wedge Q(C_2)$

## Skolemização

- Então
  - $\exists x P(x) \rightarrow P(C_1)$
  - $\exists x, y R(x, y) \rightarrow R(C_1, C_2)$  (2 variáveis)
  - $\exists x P(x) \wedge Q(x) \rightarrow P(C_1) \wedge Q(C_1)$ 
    - Cada ocorrência da variável é mapeada ao mesmo nome
  - $\exists x P(x) \wedge \exists x Q(x) \rightarrow P(C_1) \wedge Q(C_2)$ 
    - Diferentes quantificadores exigem nomes diferentes

## Skolemização

- Então

- $\exists x P(x) \rightarrow P(C_1)$
- $\exists x, y R(x, y) \rightarrow R(C_1, C_2)$  (2 variáveis)
- $\exists x P(x) \wedge Q(x) \rightarrow P(C_1) \wedge Q(C_1)$ 
  - Cada ocorrência da variável é mapeada ao mesmo nome
- $\exists x P(x) \wedge \exists x Q(x) \rightarrow P(C_1) \wedge Q(C_2)$ 
  - Diferentes quantificadores exigem nomes diferentes
  - Note que isso fica claro após a padronização:  
$$\exists x P(x) \wedge \exists x Q(x) \equiv \exists x P(x) \wedge \exists y Q(y) \rightarrow P(C_1) \wedge Q(C_2)$$

## Skolemização

- $\exists y \forall x \text{ Ama}(x, y)$

## Skolemização

- $\exists y \forall x \text{ Ama}(x, y)$   
(Existe um  $y$  que é amado por todos)



## Skolemização

- $\exists y \forall x \text{ Ama}(x, y)$   
(Existe um  $y$  que é amado por todos)
- Skolemização padrão:  $\exists y \forall x \text{ Ama}(x, y) \rightarrow \forall x \text{ Ama}(x, C_1)$

## Skolemização

- $\exists y \forall x \text{ Ama}(x, y)$   
(Existe um  $y$  que é amado por todos)
- Skolemização padrão:  $\exists y \forall x \text{ Ama}(x, y) \rightarrow \forall x \text{ Ama}(x, C_1)$
- E quanto a “ $\forall x \exists y \text{ Ama}(x, y)$ ”?

## Skolemização

- $\exists y \forall x \text{ Ama}(x, y)$   
(Existe um  $y$  que é amado por todos)
- Skolemização padrão:  $\exists y \forall x \text{ Ama}(x, y) \rightarrow \forall x \text{ Ama}(x, C_1)$
- E quanto a “ $\forall x \exists y \text{ Ama}(x, y)$ ”?  
(Todo mundo ama alguém)

## Skolemização

- $\exists y \forall x \text{ Ama}(x, y)$   
(Existe um  $y$  que é amado por todos)
- Skolemização padrão:  $\exists y \forall x \text{ Ama}(x, y) \rightarrow \forall x \text{ Ama}(x, C_1)$
- E quanto a “ $\forall x \exists y \text{ Ama}(x, y)$ ”?  
(Todo mundo ama alguém)
  - “ $\forall x \text{ Ama}(x, C_1)$ ”?

## Skolemização

- $\exists y \forall x \text{ Ama}(x, y)$   
(Existe um  $y$  que é amado por todos)
- Skolemização padrão:  $\exists y \forall x \text{ Ama}(x, y) \rightarrow \forall x \text{ Ama}(x, C_1)$
- E quanto a “ $\forall x \exists y \text{ Ama}(x, y)$ ”?  
(Todo mundo ama alguém)
  - “ $\forall x \text{ Ama}(x, C_1)$ ”?
  - Certamente não é “Todo mundo ama  $C_1$ ” que pretendíamos dizer...

## Skolemização

- $\exists y \forall x \text{ Ama}(x, y)$   
(Existe um  $y$  que é amado por todos)
- Skolemização padrão:  $\exists y \forall x \text{ Ama}(x, y) \rightarrow \forall x \text{ Ama}(x, C_1)$
- E quanto a “ $\forall x \exists y \text{ Ama}(x, y)$ ”?  
(Todo mundo ama alguém)
  - “ $\forall x \text{ Ama}(x, C_1)$ ”?
  - Certamente não é “Todo mundo ama  $C_1$ ” que pretendíamos dizer...
  - Simples substituição não funciona

## Skolemização

- O que aconteceu?

## Skolemização

- O que aconteceu?
  - Embora possamos sim dizer que “ $\forall x \exists y \text{ Ama}(x, y)$ ” se refere a um alguém específico, ele o faz para cada valor de  $x$ , ou seja, potencialmente um alguém diferente para cada  $x$



## Skolemização

- O que aconteceu?
  - Embora possamos sim dizer que “ $\forall x \exists y \text{ Ama}(x, y)$ ” se refere a um alguém específico, ele o faz para cada valor de  $x$ , ou seja, potencialmente um alguém diferente para cada  $x$
  - “ $\forall x \text{ Ama}(x, C_1)$ ” força que todo  $x$  ame um único sujeito

## Skolemização

- O que aconteceu?
  - Embora possamos sim dizer que “ $\forall x \exists y \text{ Ama}(x, y)$ ” se refere a um alguém específico, ele o faz para cada valor de  $x$ , ou seja, potencialmente um alguém diferente para cada  $x$
  - “ $\forall x \text{ Ama}(x, C_1)$ ” força que todo  $x$  ame um único sujeito
- Que fazer então?

## Skolemização

- O que aconteceu?
  - Embora possamos sim dizer que “ $\forall x \exists y \text{ Ama}(x, y)$ ” se refere a um alguém específico, ele o faz para cada valor de  $x$ , ou seja, potencialmente um alguém diferente para cada  $x$
  - “ $\forall x \text{ Ama}(x, C_1)$ ” força que todo  $x$  ame um único sujeito
- Que fazer então?
  - Fazer com que a constante utilizada dependa de  $x$

## Skolemização

- O que aconteceu?
  - Embora possamos sim dizer que “ $\forall x \exists y \text{ Ama}(x, y)$ ” se refere a um alguém específico, ele o faz para cada valor de  $x$ , ou seja, potencialmente um alguém diferente para cada  $x$
  - “ $\forall x \text{ Ama}(x, C_1)$ ” força que todo  $x$  ame um único sujeito
- Que fazer então?
  - Fazer com que a constante utilizada dependa de  $x$
  - $\forall x \exists y \text{ Ama}(x, y) \rightarrow \forall x \text{ Ama}(x, \text{AmadoPor}(x))$

## Skolemização

- O que aconteceu?
  - Embora possamos sim dizer que “ $\forall x \exists y \text{ Ama}(x, y)$ ” se refere a um alguém específico, ele o faz para cada valor de  $x$ , ou seja, potencialmente um alguém diferente para cada  $x$
  - “ $\forall x \text{ Ama}(x, C_1)$ ” força que todo  $x$  ame um único sujeito
- Que fazer então?
  - Fazer com que a constante utilizada dependa de  $x$
  - $\forall x \exists y \text{ Ama}(x, y) \rightarrow \forall x \text{ Ama}(x, \text{AmadoPor}(x))$
  - Em vez de substituímos por uma constante, substituímos por uma **Função de Skolem**

## Skolemização

- Função de Skolem – Regra geral:
  - Os argumentos da função de Skolem são todas as variáveis universalmente quantificadas em cujo escopo o quantificador existencial aparecer

## Skolemização

- Função de Skolem – Regra geral:
  - Os argumentos da função de Skolem são todas as variáveis universalmente quantificadas em cujo escopo o quantificador existencial aparecer
- Ex:

## Skolemização

- Função de Skolem – Regra geral:
  - Os argumentos da função de Skolem são todas as variáveis universalmente quantificadas em cujo escopo o quantificador existencial aparecer
- Ex:
  - $\forall x \exists y \forall z \exists w \ P(x, y, z) \wedge R(y, z, w)$   
 $\rightarrow \forall x \forall z \ P(x, F(x), z) \wedge R(F(x), z, G(x, z))$



## Skolemização

- Função de Skolem – Regra geral:
    - Os argumentos da função de Skolem são todas as variáveis universalmente quantificadas em cujo escopo o quantificador existencial aparecer
  - Ex:
    - $\forall x \exists y \forall z \exists w P(x, y, z) \wedge R(y, z, w)$   
 $\rightarrow \forall x \forall z P(x, F(x), z) \wedge R(F(x), z, G(x, z))$
- $y = F(x)$  porque  $y$  está apenas no escopo de  $\forall x$

# Lógica de Primeira Ordem – Resolução

## Skolemização

- Função de Skolem – Regra geral:
  - Os argumentos da função de Skolem são todas as variáveis universalmente quantificadas em cujo escopo o quantificador existencial aparecer
- Ex:
  - $\forall x \exists y \forall z \exists w P(x, y, z) \wedge R(y, z, w)$   
 $\rightarrow \forall x \forall z P(x, F(x), z) \wedge R(F(x), z, G(x, z))$

Já  $w = G(x, z)$  porque  
há 2 quantificadores  
universais ( $\forall x$  e  $\forall z$ )  
em cujo escopo está  $w$

# Lógica de Primeira Ordem – Resolução

## Skolemização

- Função de Skolem – Regra geral:
  - Os argumentos da função de Skolem são todas as variáveis universalmente quantificadas em cujo escopo o quantificador existencial aparecer
- Ex:
  - $\forall x \exists y \forall z \exists w P(x, y, z) \wedge R(y, z, w)$   
 $\rightarrow \forall x \forall z P(x, F(x), z) \wedge R(F(x), z, G(x, z))$ 

Já  $w = G(x, z)$  porque há 2 quantificadores universais ( $\forall x$  e  $\forall z$ ) em cujo escopo está  $w$
- A sentença Skolemizada é satisfatível exatamente quando a sentença original for satisfatível

# Lógica de Primeira Ordem – Resolução

## Convertendo para Forma Clausal: Exemplo

# Lógica de Primeira Ordem – Resolução

## Convertendo para Forma Clausal: Exemplo

- Todo mundo que ama todos os animais é amado por alguém

# Lógica de Primeira Ordem – Resolução

## Convertendo para Forma Clausal: Exemplo

- Todo mundo que ama todos os animais é amado por alguém
- $\forall x [\forall y \text{ Animal}(y) \Rightarrow \text{Ama}(x, y)] \Rightarrow [\exists y \text{ Ama}(y, x)]$

# Lógica de Primeira Ordem – Resolução

## Convertendo para Forma Clausal: Exemplo

- Todo mundo que ama todos os animais é amado por alguém
  - $\forall x [\forall y \text{ Animal}(y) \Rightarrow \text{Ama}(x, y)] \Rightarrow [\exists y \text{ Ama}(y, x)]$
- Passos:

# Lógica de Primeira Ordem – Resolução

## Convertendo para Forma Clausal: Exemplo

- Todo mundo que ama todos os animais é amado por alguém
  - $\forall x [\forall y \text{ Animal}(y) \Rightarrow \text{Ama}(x, y)] \Rightarrow [\exists y \text{ Ama}(y, x)]$
- Passos:
  - 1 Elimine implicações



# Lógica de Primeira Ordem – Resolução

## Convertendo para Forma Clausal: Exemplo

- Todo mundo que ama todos os animais é amado por alguém
  - $\forall x [\forall y \text{ Animal}(y) \Rightarrow \text{Ama}(x, y)] \Rightarrow [\exists y \text{ Ama}(y, x)]$
- Passos:
  - 1 Elimine implicações
    - $\forall x [\forall y \neg \text{Animal}(y) \vee \text{Ama}(x, y)] \Rightarrow [\exists y \text{ Ama}(y, x)]$

## Convertendo para Forma Clausal: Exemplo

- Todo mundo que ama todos os animais é amado por alguém
  - $\forall x [\forall y \text{ Animal}(y) \Rightarrow \text{Ama}(x, y)] \Rightarrow [\exists y \text{ Ama}(y, x)]$
- Passos:
  - 1 Elimine implicações
    - $\forall x [\forall y \neg \text{Animal}(y) \vee \text{Ama}(x, y)] \Rightarrow [\exists y \text{ Ama}(y, x)]$   
 $\forall x [\neg \forall y \neg \text{Animal}(y) \vee \text{Ama}(x, y)] \vee [\exists y \text{ Ama}(y, x)]$

# Lógica de Primeira Ordem – Resolução

## Convertendo para Forma Clausal: Exemplo

- Todo mundo que ama todos os animais é amado por alguém
  - $\forall x [\forall y \text{ Animal}(y) \Rightarrow \text{Ama}(x, y)] \Rightarrow [\exists y \text{ Ama}(y, x)]$
- Passos:
  - 1 Elimine implicações
    - $\forall x [\forall y \neg \text{Animal}(y) \vee \text{Ama}(x, y)] \Rightarrow [\exists y \text{ Ama}(y, x)]$   
 $\forall x [\neg \forall y \neg \text{Animal}(y) \vee \text{Ama}(x, y)] \vee [\exists y \text{ Ama}(y, x)]$
  - 2 Mova a negação “para dentro”

# Lógica de Primeira Ordem – Resolução

## Convertendo para Forma Clausal: Exemplo

- Todo mundo que ama todos os animais é amado por alguém
  - $\forall x [\forall y \text{ Animal}(y) \Rightarrow \text{Ama}(x, y)] \Rightarrow [\exists y \text{ Ama}(y, x)]$
- Passos:
  - 1 Elimine implicações
    - $\forall x [\forall y \neg \text{Animal}(y) \vee \text{Ama}(x, y)] \Rightarrow [\exists y \text{ Ama}(y, x)]$   
 $\forall x [\neg \forall y \neg \text{Animal}(y) \vee \text{Ama}(x, y)] \vee [\exists y \text{ Ama}(y, x)]$
  - 2 Mova a negação “para dentro”
    - $\forall x [\exists y \neg (\neg \text{Animal}(y) \vee \text{Ama}(x, y))] \vee [\exists y \text{ Ama}(y, x)]$

# Lógica de Primeira Ordem – Resolução

## Convertendo para Forma Clausal: Exemplo

- Todo mundo que ama todos os animais é amado por alguém

- $\forall x [\forall y \text{ Animal}(y) \Rightarrow \text{Ama}(x, y)] \Rightarrow [\exists y \text{ Ama}(y, x)]$

- Passos:

- 1 Elimine implicações

- $\forall x [\forall y \neg \text{Animal}(y) \vee \text{Ama}(x, y)] \Rightarrow [\exists y \text{ Ama}(y, x)]$

- $\forall x [\neg \forall y \neg \text{Animal}(y) \vee \text{Ama}(x, y)] \vee [\exists y \text{ Ama}(y, x)]$

- 2 Mova a negação “para dentro”

- $\forall x [\exists y \neg (\neg \text{Animal}(y) \vee \text{Ama}(x, y))] \vee [\exists y \text{ Ama}(y, x)]$

- $\forall x [\exists y \neg \neg \text{Animal}(y) \wedge \neg \text{Ama}(x, y)] \vee [\exists y \text{ Ama}(y, x)]$

# Lógica de Primeira Ordem – Resolução

## Convertendo para Forma Clausal: Exemplo

- Todo mundo que ama todos os animais é amado por alguém

- $\forall x [\forall y \text{ Animal}(y) \Rightarrow \text{Ama}(x, y)] \Rightarrow [\exists y \text{ Ama}(y, x)]$

- Passos:

- 1 Elimine implicações

- $\forall x [\forall y \neg \text{Animal}(y) \vee \text{Ama}(x, y)] \Rightarrow [\exists y \text{ Ama}(y, x)]$

$$\forall x [\neg \forall y \neg \text{Animal}(y) \vee \text{Ama}(x, y)] \vee [\exists y \text{ Ama}(y, x)]$$

- 2 Mova a negação “para dentro”

- $\forall x [\exists y \neg (\neg \text{Animal}(y) \vee \text{Ama}(x, y))] \vee [\exists y \text{ Ama}(y, x)]$

$$\forall x [\exists y \neg \neg \text{Animal}(y) \wedge \neg \text{Ama}(x, y)] \vee [\exists y \text{ Ama}(y, x)]$$

$$\forall x [\exists y \text{ Animal}(y) \wedge \neg \text{Ama}(x, y)] \vee [\exists y \text{ Ama}(y, x)]$$

## Convertendo para Forma Clausal: Exemplo

- Passos (cont.):

$$\forall x [\exists y \text{ Animal}(y) \wedge \neg \text{Ama}(x, y)] \vee [\exists y \text{ Ama}(y, x)]$$

- 3 Padronize as variáveis

## Convertendo para Forma Clausal: Exemplo

- Passos (cont.):

$$\forall x [\exists y \text{ Animal}(y) \wedge \neg \text{Ama}(x, y)] \vee [\exists y \text{ Ama}(y, x)]$$

- 3 Padronize as variáveis

- $\forall x [\exists y \text{ Animal}(y) \wedge \neg \text{Ama}(x, y)] \vee [\exists z \text{ Ama}(z, x)]$



## Convertendo para Forma Clausal: Exemplo

- Passos (cont.):

$$\forall x [\exists y \text{ Animal}(y) \wedge \neg \text{Ama}(x, y)] \vee [\exists y \text{ Ama}(y, x)]$$

- 3 Padronize as variáveis

- $\forall x [\exists y \text{ Animal}(y) \wedge \neg \text{Ama}(x, y)] \vee [\exists z \text{ Ama}(z, x)]$

- 4 Skolemize

## Convertendo para Forma Clausal: Exemplo

- Passos (cont.):

$$\forall x [\exists y \text{ Animal}(y) \wedge \neg \text{Ama}(x, y)] \vee [\exists y \text{ Ama}(y, x)]$$

- 3 Padronize as variáveis

- $\forall x [\exists y \text{ Animal}(y) \wedge \neg \text{Ama}(x, y)] \vee [\exists z \text{ Ama}(z, x)]$

- 4 Skolemize

- $\forall x [\text{Animal}(F(x)) \wedge \neg \text{Ama}(x, F(x))] \vee [\text{Ama}(G(x), x)]$

## Convertendo para Forma Clausal: Exemplo

- Passos (cont.):

$$\forall x [\exists y \text{ Animal}(y) \wedge \neg \text{Ama}(x, y)] \vee [\exists y \text{ Ama}(y, x)]$$

- 3 Padronize as variáveis

- $\forall x [\exists y \text{ Animal}(y) \wedge \neg \text{Ama}(x, y)] \vee [\exists z \text{ Ama}(z, x)]$

- 4 Skolemize

- $\forall x [\text{Animal}(F(x)) \wedge \neg \text{Ama}(x, F(x))] \vee [\text{Ama}(G(x), x)]$

$F(x)$ : o animal potencialmente não amado por  $x$

## Convertendo para Forma Clausal: Exemplo

- Passos (cont.):

$$\forall x [\exists y \text{ Animal}(y) \wedge \neg \text{Ama}(x, y)] \vee [\exists y \text{ Ama}(y, x)]$$

- 3 Padronize as variáveis

- $\forall x [\exists y \text{ Animal}(y) \wedge \neg \text{Ama}(x, y)] \vee [\exists z \text{ Ama}(z, x)]$

- 4 Skolemize

- $\forall x [\text{Animal}(F(x)) \wedge \neg \text{Ama}(x, F(x))] \vee [\text{Ama}(G(x), x)]$

$F(x)$ : o animal potencialmente não amado por  $x$

$G(x)$ : alguém que poderia amar  $x$

## Convertendo para Forma Clausal: Exemplo

- Passos (cont.):

$$\forall x [Animal(F(x)) \wedge \neg Ama(x, F(x))] \vee [Ama(G(x), x)]$$

- 5 Remova quantificadores universais

## Convertendo para Forma Clausal: Exemplo

- Passos (cont.):

$$\forall x [Animal(F(x)) \wedge \neg Ama(x, F(x))] \vee [Ama(G(x), x)]$$

- 5 Remova quantificadores universais

- $[Animal(F(x)) \wedge \neg Ama(x, F(x))] \vee [Ama(G(x), x)]$

## Convertendo para Forma Clausal: Exemplo

- Passos (cont.):

$$\forall x [Animal(F(x)) \wedge \neg Ama(x, F(x))] \vee [Ama(G(x), x)]$$

- 5 Remova quantificadores universais
  - $[Animal(F(x)) \wedge \neg Ama(x, F(x))] \vee [Ama(G(x), x)]$
  - De fato, não foram removidos, apenas tornados implícitos

## Convertendo para Forma Clausal: Exemplo

- Passos (cont.):

$$\forall x [Animal(F(x)) \wedge \neg Ama(x, F(x))] \vee [Ama(G(x), x)]$$

- 5 Remova quantificadores universais

- $[Animal(F(x)) \wedge \neg Ama(x, F(x))] \vee [Ama(G(x), x)]$

- De fato, não foram removidos, apenas tornados implícitos

- 6 Distribua  $\vee$  sobre  $\wedge$



## Convertendo para Forma Clausal: Exemplo

- Passos (cont.):

$$\forall x [Animal(F(x)) \wedge \neg Ama(x, F(x))] \vee [Ama(G(x), x)]$$

- 5 Remova quantificadores universais

- $[Animal(F(x)) \wedge \neg Ama(x, F(x))] \vee [Ama(G(x), x)]$

- De fato, não foram removidos, apenas tornados implícitos

- 6 Distribua  $\vee$  sobre  $\wedge$

- $[Animal(F(x)) \vee Ama(G(x), x)] \wedge [\neg Ama(x, F(x)) \vee Ama(G(x), x)]$

## Convertendo para Forma Clausal: Exemplo

- Passos (cont.):

$$\forall x [Animal(F(x)) \wedge \neg Ama(x, F(x))] \vee [Ama(G(x), x)]$$

- 5 Remova quantificadores universais

- $[Animal(F(x)) \wedge \neg Ama(x, F(x))] \vee [Ama(G(x), x)]$

- De fato, não foram removidos, apenas tornados implícitos

- 6 Distribua  $\vee$  sobre  $\wedge$

- $[Animal(F(x)) \vee Ama(G(x), x)] \wedge [\neg Ama(x, F(x)) \vee Ama(G(x), x)]$

cláusula

cláusula

## Convertendo para Forma Clausal: Exemplo

- Passos (cont.):

$$\forall x [Animal(F(x)) \wedge \neg Ama(x, F(x))] \vee [Ama(G(x), x)]$$

- 5 Remova quantificadores universais

- $[Animal(F(x)) \wedge \neg Ama(x, F(x))] \vee [Ama(G(x), x)]$

- De fato, não foram removidos, apenas tornados implícitos

- 6 Distribua  $\vee$  sobre  $\wedge$

- $[Animal(F(x)) \vee Ama(G(x), x)] \wedge [\neg Ama(x, F(x)) \vee Ama(G(x), x)]$

cláusula

cláusula

(Conjunção de disjunções de literais)

# Lógica de Primeira Ordem – Resolução

## Passos

### 1 Converter de LPO para forma clausal

- Generalização da Forma Normal Conjuntiva para LPO
- Uma conjunção de disjunções
- Sem quantificadores

$$\forall x \exists y P(x) \Rightarrow R(x, y)$$



$$\neg P(x) \vee R(x, F(x))$$

### 2 Determinar que variáveis substituir por quais quando da resolução

- Processo chamado Unificação

### 3 Executar a resolução

# Lógica de Primeira Ordem – Resolução

## Passos

### 1 Converter de LPO para forma clausal ✓

- Generalização da Forma Normal Conjuntiva para LPO
- Uma conjunção de disjunções
- Sem quantificadores

$$\forall x \exists y P(x) \Rightarrow R(x, y)$$



$$\neg P(x) \vee R(x, F(x))$$

### 2 Determinar que variáveis substituir por quais quando da resolução

- Processo chamado Unificação

### 3 Executar a resolução

# Lógica de Primeira Ordem – Resolução

## Passos

### 1 Converter de LPO para forma clausal ✓

- Generalização da Forma Normal Conjuntiva para LPO
- Uma conjunção de disjunções
- Sem quantificadores

$$\forall x \exists y P(x) \Rightarrow R(x, y)$$



$$\neg P(x) \vee R(x, F(x))$$

### 2 Determinar que variáveis substituir por quais quando da resolução 📎

- Processo chamado Unificação

### 3 Executar a resolução

## Unificação: Substituição

- Para entender unificação, precisamos antes do conceito de substituição

## Unificação: Substituição

- Para entender unificação, precisamos antes do conceito de substituição
- Considere a sentença atômica  $P(v_1, v_2, \dots, v_n)$



## Unificação: Substituição

- Para entender unificação, precisamos antes do conceito de substituição
- Considere a sentença atômica  $P(v_1, v_2, \dots, v_n)$
- Uma **substituição** é um mapeamento finito de variáveis a termos

$$\{v_1/t_1, v_2/t_2, \dots, v_n/t_n\}$$

em que

## Unificação: Substituição

- Para entender unificação, precisamos antes do conceito de substituição
- Considere a sentença atômica  $P(v_1, v_2, \dots, v_n)$
- Uma **substituição** é um mapeamento finito de variáveis a termos

$$\{v_1/t_1, v_2/t_2, \dots, v_n/t_n\}$$

em que

- Toda e qualquer variável  $v_i$  é substituída pelo termo  $t_i$  (que pode ser uma constante, outra variável, ou função)

## Unificação: Substituição

- Para entender unificação, precisamos antes do conceito de substituição
- Considere a sentença atômica  $P(v_1, v_2, \dots, v_n)$
- Uma **substituição** é um mapeamento finito de variáveis a termos

$$\{v_1/t_1, v_2/t_2, \dots, v_n/t_n\}$$

em que

- Toda e qualquer variável  $v_i$  é substituída pelo termo  $t_i$  (que pode ser uma constante, outra variável, ou função)
- Não pode haver mais de uma substituição para cada variável

# Lógica de Primeira Ordem – Resolução

## Unificação: Substituição

- Ex:  $P(x, f(y), B)$

<i>Substituição</i>	<i>Resultado</i>	<i>Obs</i>

# Lógica de Primeira Ordem – Resolução

## Unificação: Substituição

- Ex:  $P(x, f(y), B)$

<i>Substituição</i>	<i>Resultado</i>	<i>Obs</i>
$\{x/z, y/w\}$		

# Lógica de Primeira Ordem – Resolução

## Unificação: Substituição

- Ex:  $P(x, f(y), B)$

<i>Substituição</i>	<i>Resultado</i>	<i>Obs</i>
$\{x/z, y/w\}$	$P(z, f(w), B)$	

# Lógica de Primeira Ordem – Resolução

## Unificação: Substituição

- Ex:  $P(x, f(y), B)$

<i>Substituição</i>	<i>Resultado</i>	<i>Obs</i>
$\{x/z, y/w\}$	$P(z, f(w), B)$	Variação alfabética

# Lógica de Primeira Ordem – Resolução

## Unificação: Substituição

- Ex:  $P(x, f(y), B)$

<i>Substituição</i>	<i>Resultado</i>	<i>Obs</i>
$\{x/z, y/w\}$	$P(z, f(w), B)$	Variação alfabética
$\{y/A\}$		



# Lógica de Primeira Ordem – Resolução

## Unificação: Substituição

- Ex:  $P(x, f(y), B)$

<i>Substituição</i>	<i>Resultado</i>	<i>Obs</i>
$\{x/z, y/w\}$	$P(z, f(w), B)$	Variação alfabética
$\{y/A\}$	$P(x, f(A), B)$	

# Lógica de Primeira Ordem – Resolução

## Unificação: Substituição

- Ex:  $P(x, f(y), B)$

<i>Substituição</i>	<i>Resultado</i>	<i>Obs</i>
$\{x/z, y/w\}$	$P(z, f(w), B)$	Variação alfabética
$\{y/A\}$	$P(x, f(A), B)$	
$\{x/g(z), y/A\}$		

# Lógica de Primeira Ordem – Resolução

## Unificação: Substituição

- Ex:  $P(x, f(y), B)$

<i>Substituição</i>	<i>Resultado</i>	<i>Obs</i>
$\{x/z, y/w\}$	$P(z, f(w), B)$	Variação alfabética
$\{y/A\}$	$P(x, f(A), B)$	
$\{x/g(z), y/A\}$	$P(g(z), f(A), B)$	

# Lógica de Primeira Ordem – Resolução

## Unificação: Substituição

- Ex:  $P(x, f(y), B)$

<i>Substituição</i>	<i>Resultado</i>	<i>Obs</i>
$\{x/z, y/w\}$	$P(z, f(w), B)$	Variação alfabética
$\{y/A\}$	$P(x, f(A), B)$	
$\{x/g(z), y/A\}$	$P(g(z), f(A), B)$	
$\{x/C, y/A\}$		

# Lógica de Primeira Ordem – Resolução

## Unificação: Substituição

- Ex:  $P(x, f(y), B)$

<i>Substituição</i>	<i>Resultado</i>	<i>Obs</i>
$\{x/z, y/w\}$	$P(z, f(w), B)$	Variação alfabética
$\{y/A\}$	$P(x, f(A), B)$	
$\{x/g(z), y/A\}$	$P(g(z), f(A), B)$	
$\{x/C, y/A\}$	$P(C, f(A), B)$	

# Lógica de Primeira Ordem – Resolução

## Unificação: Substituição

- Ex:  $P(x, f(y), B)$

<i>Substituição</i>	<i>Resultado</i>	<i>Obs</i>
$\{x/z, y/w\}$	$P(z, f(w), B)$	Variação alfabética
$\{y/A\}$	$P(x, f(A), B)$	
$\{x/g(z), y/A\}$	$P(g(z), f(A), B)$	
$\{x/C, y/A\}$	$P(C, f(A), B)$	Sentença constante

# Lógica de Primeira Ordem – Resolução

## Unificação: Substituição

- Ex:  $P(x, f(y), B)$

<i>Substituição</i>	<i>Resultado</i>	<i>Obs</i>
$\{x/z, y/w\}$	$P(z, f(w), B)$	Variação alfabética
$\{y/A\}$	$P(x, f(A), B)$	
$\{x/g(z), y/A\}$	$P(g(z), f(A), B)$	
$\{x/C, y/A\}$	$P(C, f(A), B)$	Sentença constante
$\{y/A, x/y\}$		

# Lógica de Primeira Ordem – Resolução

## Unificação: Substituição

- Ex:  $P(x, f(y), B)$

<i>Substituição</i>	<i>Resultado</i>	<i>Obs</i>
$\{x/z, y/w\}$	$P(z, f(w), B)$	Variação alfabética
$\{y/A\}$	$P(x, f(A), B)$	
$\{x/g(z), y/A\}$	$P(g(z), f(A), B)$	
$\{x/C, y/A\}$	$P(C, f(A), B)$	Sentença constante
$\{y/A, x/y\}$	$P(A, f(A), B)$	



# Lógica de Primeira Ordem – Resolução

## Unificação: Substituição

- Ex:  $P(x, f(y), B)$

<i>Substituição</i>	<i>Resultado</i>	<i>Obs</i>
$\{x/z, y/w\}$	$P(z, f(w), B)$	Variação alfabética
$\{y/A\}$	$P(x, f(A), B)$	
$\{x/g(z), y/A\}$	$P(g(z), f(A), B)$	
$\{x/C, y/A\}$	$P(C, f(A), B)$	Sentença constante
$\{y/A, x/y\}$	$P(A, f(A), B)$	Sentença constante

# Lógica de Primeira Ordem – Resolução

## Unificação: Substituição

- Ex:  $P(x, f(y), B)$

<i>Substituição</i>	<i>Resultado</i>	<i>Obs</i>
$\{x/z, y/w\}$	$P(z, f(w), B)$	Variação alfabética
$\{y/A\}$	$P(x, f(A), B)$	
$\{x/g(z), y/A\}$	$P(g(z), f(A), B)$	
$\{x/C, y/A\}$	$P(C, f(A), B)$	Sentença constante
$\{y/A, x/y\}$	$P(A, f(A), B)$	Sentença constante

- Variação alfabética: Quando apenas substituímos por variáveis diferentes

# Lógica de Primeira Ordem – Resolução

## Unificação: Substituição

- Ex:  $P(x, f(y), B)$

<i>Substituição</i>	<i>Resultado</i>	<i>Obs</i>
$\{x/z, y/w\}$	$P(z, f(w), B)$	Variação alfabética
$\{y/A\}$	$P(x, f(A), B)$	
$\{x/g(z), y/A\}$	$P(g(z), f(A), B)$	
$\{x/C, y/A\}$	$P(C, f(A), B)$	Sentença constante
$\{y/A, x/y\}$	$P(A, f(A), B)$	Sentença constante

- Variação alfabética: Quando apenas substituímos por variáveis diferentes
- Sentença constante (Ground instance): Sentença atômica sem variáveis

## Unificação: Substituição

- Sentenças com substituições são mais específicas

## Unificação: Substituição

- Sentenças com substituições são mais específicas
  - Possuem menos interpretações sob as quais são verdadeiras

## Unificação: Substituição

- Sentenças com substituições são mais específicas
  - Possuem menos interpretações sob as quais são verdadeiras
  - Ex:  $P(g(z), f(A), B)$  é mais específica que  $P(x, f(y), B)$

## Unificação: Substituição

- Sentenças com substituições são mais específicas
  - Possuem menos interpretações sob as quais são verdadeiras
  - Ex:  $P(g(z), f(A), B)$  é mais específica que  $P(x, f(y), B)$
- Não é permitido substituir constantes ou termos compostos (funções etc)

## Unificação: Substituição

- Sentenças com substituições são mais específicas
  - Possuem menos interpretações sob as quais são verdadeiras
  - Ex:  $P(g(z), f(A), B)$  é mais específica que  $P(x, f(y), B)$
- Não é permitido substituir constantes ou termos compostos (funções etc)
  - Podemos apenas substituir as variáveis, mesmo que estas estejam dentro de um termo composto (como o  $y$  em  $f(y)$ )



## Unificação: Substituição

- Não confunda com interpretação semântica:

## Unificação: Substituição

- Não confunda com interpretação semântica:
  - A substituição troca uma variável por um termo (sintaxe) de modo a produzir uma nova sentença

## Unificação: Substituição

- Não confunda com interpretação semântica:
  - A substituição troca uma variável por um termo (sintaxe) de modo a produzir uma nova sentença
  - A interpretação mapeia a variável a um objeto no domínio (semântica)

## Unificação

- Trata-se de encontrar uma substituição que faça com que duas expressões se igualem

## Unificação

- Trata-se de encontrar uma substituição que faça com que duas expressões se igualem
- Estamos interessados em modos de fazer expressões se equivalerem, em toda interpretação de seus símbolos

## Unificação

- Trata-se de encontrar uma substituição que faça com que duas expressões se igualem
  - Estamos interessados em modos de fazer expressões se equivalerem, em toda interpretação de seus símbolos
- Expressões unificáveis:

## Unificação

- Trata-se de encontrar uma substituição que faça com que duas expressões se igualem
  - Estamos interessados em modos de fazer expressões se equivalerem, em toda interpretação de seus símbolos
- Expressões unificáveis:
  - $\omega_1$  e  $\omega_2$  são unificáveis sse existir uma substituição  $s$  tal que  $\omega_1 s = \omega_2 s$

## Unificação

- Trata-se de encontrar uma substituição que faça com que duas expressões se igualem
- Estamos interessados em modos de fazer expressões se equivalerem, em toda interpretação de seus símbolos
- Expressões unificáveis:
  - $\omega_1$  e  $\omega_2$  são unificáveis sse existir uma substituição  $s$  tal que  $\omega_1 s = \omega_2 s$
  - Se existir uma substituição  $s$  que, quando aplicada a  $\omega_1$  e  $\omega_2$ , nos leva à mesma expressão



## Unificação

- Trata-se de encontrar uma substituição que faça com que duas expressões se igualem
  - Estamos interessados em modos de fazer expressões se equivalerem, em toda interpretação de seus símbolos
- Expressões unificáveis:
  - $\omega_1$  e  $\omega_2$  são unificáveis sse existir uma substituição  $s$  tal que  $\omega_1 s = \omega_2 s$ 
    - Se existir uma substituição  $s$  que, quando aplicada a  $\omega_1$  e  $\omega_2$ , nos leva à mesma expressão
  - $s$  é então um **unificador** de  $\omega_1$  e  $\omega_2$

## Unificação

- Ex:  $\omega_1 = x$  e  $\omega_2 = y$

## Unificação

- Ex:  $\omega_1 = x$  e  $\omega_2 = y$
- Unificadores:

$s$	$\omega_1 s$	$\omega_2 s$

## Unificação

- Ex:  $\omega_1 = x$  e  $\omega_2 = y$
- Unificadores:

$s$	$\omega_1 s$	$\omega_2 s$
$\{y/x\}$		

## Unificação

- Ex:  $\omega_1 = x$  e  $\omega_2 = y$
- Unificadores:

$s$	$\omega_1 s$	$\omega_2 s$
$\{y/x\}$	$x$	

## Unificação

- Ex:  $\omega_1 = x$  e  $\omega_2 = y$
- Unificadores:

$s$	$\omega_1 s$	$\omega_2 s$
$\{y/x\}$	$x$	$x$

## Unificação

- Ex:  $\omega_1 = x$  e  $\omega_2 = y$
- Unificadores:

$s$	$\omega_1 s$	$\omega_2 s$
$\{y/x\}$	$x$	$x$
$\{x/y\}$		

## Unificação

- Ex:  $\omega_1 = x$  e  $\omega_2 = y$
- Unificadores:

$s$	$\omega_1 s$	$\omega_2 s$
$\{y/x\}$	$x$	$x$
$\{x/y\}$	$y$	



## Unificação

- Ex:  $\omega_1 = x$  e  $\omega_2 = y$
- Unificadores:

$s$	$\omega_1 s$	$\omega_2 s$
$\{y/x\}$	$x$	$x$
$\{x/y\}$	$y$	$y$

# Lógica de Primeira Ordem – Resolução

## Unificação

- Ex:  $\omega_1 = x$  e  $\omega_2 = y$
- Unificadores:

$s$	$\omega_1 s$	$\omega_2 s$
$\{y/x\}$	$x$	$x$
$\{x/y\}$	$y$	$y$
$\{x/f(f(A)), y/f(f(A))\}$		

# Lógica de Primeira Ordem – Resolução

## Unificação

- Ex:  $\omega_1 = x$  e  $\omega_2 = y$
- Unificadores:

$s$	$\omega_1 s$	$\omega_2 s$
$\{y/x\}$	$x$	$x$
$\{x/y\}$	$y$	$y$
$\{x/f(f(A)), y/f(f(A))\}$	$f(f(A))$	

# Lógica de Primeira Ordem – Resolução

## Unificação

- Ex:  $\omega_1 = x$  e  $\omega_2 = y$
- Unificadores:

$s$	$\omega_1 s$	$\omega_2 s$
$\{y/x\}$	$x$	$x$
$\{x/y\}$	$y$	$y$
$\{x/f(f(A)), y/f(f(A))\}$	$f(f(A))$	$f(f(A))$

# Lógica de Primeira Ordem – Resolução

## Unificação

- Ex:  $\omega_1 = x$  e  $\omega_2 = y$
- Unificadores:

$s$	$\omega_1 s$	$\omega_2 s$
$\{y/x\}$	$x$	$x$
$\{x/y\}$	$y$	$y$
$\{x/f(f(A)), y/f(f(A))\}$	$f(f(A))$	$f(f(A))$
$\{x/A, y/A\}$		

# Lógica de Primeira Ordem – Resolução

## Unificação

- Ex:  $\omega_1 = x$  e  $\omega_2 = y$
- Unificadores:

$s$	$\omega_1 s$	$\omega_2 s$
$\{y/x\}$	$x$	$x$
$\{x/y\}$	$y$	$y$
$\{x/f(f(A)), y/f(f(A))\}$	$f(f(A))$	$f(f(A))$
$\{x/A, y/A\}$	$A$	

# Lógica de Primeira Ordem – Resolução

## Unificação

- Ex:  $\omega_1 = x$  e  $\omega_2 = y$
- Unificadores:

$s$	$\omega_1 s$	$\omega_2 s$
$\{y/x\}$	$x$	$x$
$\{x/y\}$	$y$	$y$
$\{x/f(f(A)), y/f(f(A))\}$	$f(f(A))$	$f(f(A))$
$\{x/A, y/A\}$	$A$	$A$

# Lógica de Primeira Ordem – Resolução

## Unificação

- Ex:  $\omega_1 = x$  e  $\omega_2 = y$
- Unificadores:

$s$	$\omega_1 s$	$\omega_2 s$
$\{y/x\}$	$x$	$x$
$\{x/y\}$	$y$	$y$
$\{x/f(f(A)), y/f(f(A))\}$	$f(f(A))$	$f(f(A))$
$\{x/A, y/A\}$	$A$	$A$

- Pode haver vários unificadores para o mesmo par de expressões, alguns mais gerais que outros



# Lógica de Primeira Ordem – Resolução

## Unificação

- Ex:  $\omega_1 = x$  e  $\omega_2 = y$
- Unificadores:

$s$	$\omega_1 s$	$\omega_2 s$
$\{y/x\}$	$x$	$x$
$\{x/y\}$	$y$	$y$
$\{x/f(f(A)), y/f(f(A))\}$	$f(f(A))$	$f(f(A))$
$\{x/A, y/A\}$	$A$	$A$

- Pode haver vários unificadores para o mesmo par de expressões, alguns mais gerais que outros
- Ex:  $\{y/x\}$  é mais geral que  $\{x/A, y/A\}$ , por impor menos restrições aos valores das variáveis

## Unificação

- Para um algoritmo computacional, seria desejável possuir uma única substituição – um único unificador

## Unificação

- Para um algoritmo computacional, seria desejável possuir uma única substituição – um único unificador
- Ocorre que, para todo par de expressões unificáveis, há um único **unificador mais geral**

## Unificação

- Para um algoritmo computacional, seria desejável possuir uma única substituição – um único unificador
- Ocorre que, para todo par de expressões unificáveis, há um único **unificador mais geral**
  - Este é único até o ponto de renomeação de variáveis

## Unificação

- Para um algoritmo computacional, seria desejável possuir uma única substituição – um único unificador
- Ocorre que, para todo par de expressões unificáveis, há um único **unificador mais geral**
  - Este é único até o ponto de renomeação de variáveis
  - Ex:  $\{x/Pedro\}$  e  $\{y/Pedro\}$  são consideradas equivalentes

## Unificação

- Para um algoritmo computacional, seria desejável possuir uma única substituição – um único unificador
- Ocorre que, para todo par de expressões unificáveis, há um único **unificador mais geral**
  - Este é único até o ponto de renomeação de variáveis
  - Ex:  $\{x/Pedro\}$  e  $\{y/Pedro\}$  são consideradas equivalentes
    - Só diferem pelo nome das variáveis

## Unificação

- Para um algoritmo computacional, seria desejável possuir uma única substituição – um único unificador
- Ocorre que, para todo par de expressões unificáveis, há um único **unificador mais geral**
  - Este é único até o ponto de renomeação de variáveis
  - Ex:  $\{x/Pedro\}$  e  $\{y/Pedro\}$  são consideradas equivalentes
    - Só diferem pelo nome das variáveis
  - Assim como  $\{x/Pedro, y/x\}$  e  $\{y/Pedro, x/z\}$

## Unificador mais Geral (UMG)

- $g$  é um unificador mais geral de  $\omega_1$  e  $\omega_2$  sse, para todo unificador  $s$ , houver um  $s'$  tal que



## Unificador mais Geral (UMG)

- $g$  é um unificador mais geral de  $\omega_1$  e  $\omega_2$  sse, para todo unificador  $s$ , houver um  $s'$  tal que
  - $\omega_1 s = (\omega_1 g) s'$ , e

## Unificador mais Geral (UMG)

- $g$  é um unificador mais geral de  $\omega_1$  e  $\omega_2$  sse, para todo unificador  $s$ , houver um  $s'$  tal que
  - $\omega_1 s = (\omega_1 g) s'$ , e
  - $\omega_2 s = (\omega_2 g) s'$

## Unificador mais Geral (UMG)

- $g$  é um unificador mais geral de  $\omega_1$  e  $\omega_2$  sse, para todo unificador  $s$ , houver um  $s'$  tal que
  - $\omega_1 s = (\omega_1 g) s'$ , e
  - $\omega_2 s = (\omega_2 g) s'$
- Ou seja, um unificador  $g$  é o mais geral se todo outro unificador  $s$  puder ser expressado como uma substituição extra  $s'$  adicionada ao mais geral

## Unificador mais Geral (UMG)

- $g$  é um unificador mais geral de  $\omega_1$  e  $\omega_2$  sse, para todo unificador  $s$ , houver um  $s'$  tal que
  - $\omega_1 s = (\omega_1 g) s'$ , e
  - $\omega_2 s = (\omega_2 g) s'$
- Ou seja, um unificador  $g$  é o mais geral se todo outro unificador  $s$  puder ser expressado como uma substituição extra  $s'$  adicionada ao mais geral
  - É então a substituição mais geral capaz de tornar as duas expressões iguais

# Lógica de Primeira Ordem – Resolução

## Unificador mais Geral (UMG): Exemplo

$\omega_1$	$\omega_2$	$UMG(\sigma)$	$\omega'_1$ e $\omega'_2$
$P(x)$	$P(A)$	$\{x/A\}$	
$P(f(x), y, g(x))$	$P(f(x), x, g(x))$	$\{y/x\}$ (ou $\{x/y\}$ )	
$P(f(x), y, g(y))$	$P(f(x), z, g(x))$	$\{y/x, z/x\}$	
$P(x, B, B)$	$P(A, y, z)$	$\{x/A, y/B, z/B\}$	
$P(g(f(v)), g(u))$	$P(x, x)$	$\{x/g(f(v)), u/f(v)\}$	
$P(x, f(x))$	$P(x, x)$		

# Lógica de Primeira Ordem – Resolução

## Unificador mais Geral (UMG): Exemplo

$\omega_1$	$\omega_2$	$UMG(\sigma)$	$\omega'_1$ e $\omega'_2$
$P(x)$	$P(A)$	$\{x/A\}$	$P(A)$
$P(f(x), y, g(x))$	$P(f(x), x, g(x))$	$\{y/x\}$ (ou $\{x/y\}$ )	
$P(f(x), y, g(y))$	$P(f(x), z, g(x))$	$\{y/x, z/x\}$	
$P(x, B, B)$	$P(A, y, z)$	$\{x/A, y/B, z/B\}$	
$P(g(f(v)), g(u))$	$P(x, x)$	$\{x/g(f(v)), u/f(v)\}$	
$P(x, f(x))$	$P(x, x)$		

# Lógica de Primeira Ordem – Resolução

## Unificador mais Geral (UMG): Exemplo

$\omega_1$	$\omega_2$	$UMG(\sigma)$	$\omega'_1$ e $\omega'_2$
$P(x)$	$P(A)$	$\{x/A\}$	$P(A)$
$P(f(x), y, g(x))$	$P(f(x), x, g(x))$	$\{y/x\}$ (ou $\{x/y\}$ )	$P(f(x), x, g(x))$
$P(f(x), y, g(y))$	$P(f(x), z, g(x))$	$\{y/x, z/x\}$	
$P(x, B, B)$	$P(A, y, z)$	$\{x/A, y/B, z/B\}$	
$P(g(f(v)), g(u))$	$P(x, x)$	$\{x/g(f(v)), u/f(v)\}$	
$P(x, f(x))$	$P(x, x)$		

# Lógica de Primeira Ordem – Resolução

## Unificador mais Geral (UMG): Exemplo

$\omega_1$	$\omega_2$	$UMG(\sigma)$	$\omega'_1$ e $\omega'_2$
$P(x)$	$P(A)$	$\{x/A\}$	$P(A)$
$P(f(x), y, g(x))$	$P(f(x), x, g(x))$	$\{y/x\}$ (ou $\{x/y\}$ )	$P(f(x), x, g(x))$
$P(f(x), y, g(y))$	$P(f(x), z, g(x))$	$\{y/x, z/x\}$	$P(f(x), x, g(x))$
$P(x, B, B)$	$P(A, y, z)$	$\{x/A, y/B, z/B\}$	
$P(g(f(v)), g(u))$	$P(x, x)$	$\{x/g(f(v)), u/f(v)\}$	
$P(x, f(x))$	$P(x, x)$		



# Lógica de Primeira Ordem – Resolução

## Unificador mais Geral (UMG): Exemplo

$\omega_1$	$\omega_2$	$UMG(\sigma)$	$\omega'_1$ e $\omega'_2$
$P(x)$	$P(A)$	$\{x/A\}$	$P(A)$
$P(f(x), y, g(x))$	$P(f(x), x, g(x))$	$\{y/x\}$ (ou $\{x/y\}$ )	$P(f(x), x, g(x))$
$P(f(x), y, g(y))$	$P(f(x), z, g(x))$	$\{y/x, z/x\}$	$P(f(x), x, g(x))$
$P(x, B, B)$	$P(A, y, z)$	$\{x/A, y/B, z/B\}$	$P(A, B, B)$
$P(g(f(v)), g(u))$	$P(x, x)$	$\{x/g(f(v)), u/f(v)\}$	
$P(x, f(x))$	$P(x, x)$		

# Lógica de Primeira Ordem – Resolução

## Unificador mais Geral (UMG): Exemplo

$\omega_1$	$\omega_2$	$UMG(\sigma)$	$\omega'_1$ e $\omega'_2$
$P(x)$	$P(A)$	$\{x/A\}$	$P(A)$
$P(f(x), y, g(x))$	$P(f(x), x, g(x))$	$\{y/x\}$ (ou $\{x/y\}$ )	$P(f(x), x, g(x))$
$P(f(x), y, g(y))$	$P(f(x), z, g(x))$	$\{y/x, z/x\}$	$P(f(x), x, g(x))$
$P(x, B, B)$	$P(A, y, z)$	$\{x/A, y/B, z/B\}$	$P(A, B, B)$
$P(g(f(v)), g(u))$	$P(x, x)$	$\{x/g(f(v)), u/f(v)\}$	$P(g(f(v)), g(f(v)))$
$P(x, f(x))$	$P(x, x)$		

# Lógica de Primeira Ordem – Resolução

## Unificador mais Geral (UMG): Exemplo

$\omega_1$	$\omega_2$	$UMG(\sigma)$	$\omega'_1$ e $\omega'_2$
$P(x)$	$P(A)$	$\{x/A\}$	$P(A)$
$P(f(x), y, g(x))$	$P(f(x), x, g(x))$	$\{y/x\}$ (ou $\{x/y\}$ )	$P(f(x), x, g(x))$
$P(f(x), y, g(y))$	$P(f(x), z, g(x))$	$\{y/x, z/x\}$	$P(f(x), x, g(x))$
$P(x, B, B)$	$P(A, y, z)$	$\{x/A, y/B, z/B\}$	$P(A, B, B)$
$P(g(f(v)), g(u))$	$P(x, x)$	$\{x/g(f(v)), u/f(v)\}$	$P(g(f(v)), g(f(v)))$
$P(x, f(x))$	$P(x, x)$	Não há	

# Lógica de Primeira Ordem – Resolução

## Unificador mais Geral (UMG): Funcionamento

## Unificador mais Geral (UMG): Funcionamento

- O algoritmo recursivamente compara a estrutura de duas expressões,  $x$  e  $y$ , elemento por elemento

## Unificador mais Geral (UMG): Funcionamento

- O algoritmo recursivamente compara a estrutura de duas expressões,  $x$  e  $y$ , elemento por elemento
  - Ao longo do caminho, constrói um unificador  $s$  para as expressões

## Unificador mais Geral (UMG): Funcionamento

- O algoritmo recursivamente compara a estrutura de duas expressões,  $x$  e  $y$ , elemento por elemento
  - Ao longo do caminho, constrói um unificador  $s$  para as expressões
  - Falha se dois pontos correspondentes nas estruturas não baterem

## Unificador mais Geral (UMG): Funcionamento

- O algoritmo recursivamente compara a estrutura de duas expressões,  $x$  e  $y$ , elemento por elemento
  - Ao longo do caminho, constrói um unificador  $s$  para as expressões
  - Falha se dois pontos correspondentes nas estruturas não baterem
  - Usa assim  $s$  para garantir que comparações futuras sejam consistentes com ligações estabelecidas anteriormente



## Unificador mais Geral (UMG): Funcionamento

- Ao comparar uma variável com um termo complexo, o algoritmo faz a **checagem de ocorrência**:

## Unificador mais Geral (UMG): Funcionamento

- Ao comparar uma variável com um termo complexo, o algoritmo faz a **checagem de ocorrência**:
  - Verifica se essa variável já não ocorre dentro desse termo

## Unificador mais Geral (UMG): Funcionamento

- Ao comparar uma variável com um termo complexo, o algoritmo faz a **checagem de ocorrência**:
  - Verifica se essa variável já não ocorre dentro desse termo
  - Se sim, falha, pois nenhum unificador consistente pode ser construído

## Unificador mais Geral (UMG): Funcionamento

- Ao comparar uma variável com um termo complexo, o algoritmo faz a **checagem de ocorrência**:
  - Verifica se essa variável já não ocorre dentro desse termo
  - Se sim, falha, pois nenhum unificador consistente pode ser construído
  - Ex:  $S(x)$  não pode ser unificado a  $S(S(x))$

# Lógica de Primeira Ordem – Resolução

## Unificador mais Geral (UMG): Algoritmo

**Função** *Unifica*( $x, y, s$ ): substituição

**se**  $s = \text{falha}$  **então retorna** *falha*

**se**  $x = y$  **então retorna**  $s$

**se**  $x$  *for uma variável* **então**

└ **retorna** *UnificaVar*( $x, y, s$ )

**se**  $y$  *for uma variável* **então**

└ **retorna** *UnificaVar*( $y, x, s$ )

**se**  $x$  e  $y$  *forem expressões compostas* **então**

└ **retorna** *Unifica*(*args*( $x$ ), *args*( $y$ ), *Unifica*(*op*( $x$ ), *op*( $y$ ),  $s$ ))

**se**  $x$  e  $y$  *forem listas* **então**

└ **retorna** *Unifica*(*resto*( $x$ ), *resto*( $y$ ), *Unifica*(*primeiro*( $x$ ), *primeiro*( $y$ ),  $s$ ))

└ **retorna** *falha*

# Lógica de Primeira Ordem – Resolução

## Unificador mais Geral (UMG): Algoritmo

**Função** *Unifica*(*x,y,s*): substituição

**se** *s* = *falha* **então retorna** *falha*

**se** *x* = *y* **então retorna** *s*

**se** *x* *for uma variável* **então**

└ **retorna** *UnificaVar*(*x,y,s*)

**se** *y* *for uma variável* **então**

└ **retorna** *UnificaVar*(*y,x,s*)

**se** *x* e *y* *forem expressões compostas* **então**

└ **retorna** *Unifica*(*args*(*x*), *args*(*y*), *Unifica*(*op*(*x*), *op*(*y*), *s*))

**se** *x* e *y* *forem listas* **então**

└ **retorna** *Unifica*(*resto*(*x*), *resto*(*y*), *Unifica*(*primeiro*(*x*),  
*primeiro*(*y*), *s*))

└ **retorna** *falha*

Variável, constante, lista  
ou expressão composta

# Lógica de Primeira Ordem – Resolução

## Unificador mais Geral (UMG): Algoritmo

**Função** *Unifica*(*x*, *y*, *s*): **substituição**

**se** *s* = *falha* **então retorna** *falha*

**se** *x* = *y* **então retorna** *s*

**se** *x* *for uma variável* **então**

└ **retorna** *UnificaVar*(*x*, *y*, *s*)

**se** *y* *for uma variável* **então**

└ **retorna** *UnificaVar*(*y*, *x*, *s*)

**se** *x* e *y* *forem expressões compostas* **então**

└ **retorna** *Unifica*(*args*(*x*), *args*(*y*), *Unifica*(*op*(*x*), *op*(*y*), *s*))

**se** *x* e *y* *forem listas* **então**

└ **retorna** *Unifica*(*resto*(*x*), *resto*(*y*), *Unifica*(*primeiro*(*x*),  
*primeiro*(*y*), *s*))

└ **retorna** *falha*

A substituição construída  
até então (no início, vazia)

# Lógica de Primeira Ordem – Resolução

## Unificador mais Geral (UMG): Algoritmo

**Função**  $Unifica(x,y,s)$ : **substituição**

se  $s = falha$  então retorna  $falha$

se  $x = y$  então retorna  $s$

se  $x$  for uma variável então

└ retorna  $UnificaVar(x,y,s)$

se  $y$  for uma variável então

└ retorna  $UnificaVar(y,x,s)$

se  $x$  e  $y$  forem expressões compostas então

└ retorna  $Unifica(args(x), args(y), Unifica(op(x), op(y), s))$

se  $x$  e  $y$  forem listas então

└ retorna  $Unifica(resto(x), resto(y), Unifica(primeiro(x), primeiro(y), s))$

└ retorna  $falha$

Retorna uma substituição, se  $x$  e  $y$  forem unificáveis no contexto de  $s$ . Do contrário, falha



# Lógica de Primeira Ordem – Resolução

## Unificador mais Geral (UMG): Algoritmo

**Função** *Unifica*(*x*,*y*,*s*): substituição

**se** *s* = *falha* **então retorna** *falha*

**se** *x* = *y* **então retorna** *s* ←

**se** *x* *for uma variável* **então**

└ **retorna** *UnificaVar*(*x*,*y*,*s*)

**se** *y* *for uma variável* **então**

└ **retorna** *UnificaVar*(*y*,*x*,*s*)

**se** *x* e *y* *forem expressões compostas* **então**

└ **retorna** *Unifica*(*args*(*x*), *args*(*y*), *Unifica*(*op*(*x*),*op*(*y*),*s*))

**se** *x* e *y* *forem listas* **então**

└ **retorna** *Unifica*(*resto*(*x*), *resto*(*y*), *Unifica*(*primeiro*(*x*),  
*primeiro*(*y*), *s*))

└ **retorna** *falha*

*x* e *y* contém valores ou expressões idênticos (esses podem ter sido atribuídos a elas em *UnificaVar*)

# Lógica de Primeira Ordem – Resolução

## Unificador mais Geral (UMG): Algoritmo

**Função**  $Unifica(x,y,s)$ : substituição

**se**  $s = falha$  **então retorna**  $falha$

**se**  $x = y$  **então retorna**  $s$

**se**  $x$  *for uma variável* **então**  
└ **retorna**  $UnificaVar(x,y,s)$

← Se  $x$  ou  $y$  forem variáveis, tenta unificá-las à outra expressão

**se**  $y$  *for uma variável* **então**  
└ **retorna**  $UnificaVar(y,x,s)$

**se**  $x$  e  $y$  *forem expressões compostas* **então**  
└ **retorna**  $Unifica(args(x), args(y), Unifica(op(x), op(y), s))$

**se**  $x$  e  $y$  *forem listas* **então**  
└ **retorna**  $Unifica(resto(x), resto(y), Unifica(primeiro(x), primeiro(y), s))$

└ **retorna**  $falha$

# Lógica de Primeira Ordem – Resolução

## Unificador mais Geral (UMG): Algoritmo

**Função** *Unifica*( $x, y, s$ ): substituição

**se**  $s = \text{falha}$  **então retorna** *falha*

**se**  $x = y$  **então retorna**  $s$

**se**  $x$  *for uma variável* **então**

└ **retorna** *UnificaVar*( $x, y, s$ ) ←

*UnificaVar* tenta unificar a variável do primeiro argumento com a expressão do segundo, no contexto de  $s$

**se**  $y$  *for uma variável* **então**

└ **retorna** *UnificaVar*( $y, x, s$ )

**se**  $x$  e  $y$  *forem expressões compostas* **então**

└ **retorna** *Unifica*( $\text{args}(x), \text{args}(y), \text{Unifica}(\text{op}(x), \text{op}(y), s)$ )

**se**  $x$  e  $y$  *forem listas* **então**

└ **retorna** *Unifica*( $\text{resto}(x), \text{resto}(y), \text{Unifica}(\text{primeiro}(x), \text{primeiro}(y), s)$ )

└ **retorna** *falha*

# Lógica de Primeira Ordem – Resolução

## Unificador mais Geral (UMG): Algoritmo

**Função** *Unifica*( $x, y, s$ ): substituição

se  $s = \text{falha}$  então retorna *falha*

se  $x = y$  então retorna  $s$

se  $x$  for uma variável então

└ retorna *UnificaVar*( $x, y, s$ )

se  $y$  for uma variável então

└ retorna *UnificaVar*( $y, x, s$ )

se  $x$  e  $y$  forem *expressões compostas* então

└ retorna *Unifica*(*args*( $x$ ), *args*( $y$ ), *Unifica*(*op*( $x$ ), *op*( $y$ ),  $s$ ))

se  $x$  e  $y$  forem listas então

└ retorna *Unifica*(*resto*( $x$ ), *resto*( $y$ ), *Unifica*(*primeiro*( $x$ ),  
└ *primeiro*( $y$ ),  $s$ ))

└ retorna *falha*

Em uma expressão composta, como  $F(A, B)$ , *op*() captura o símbolo de função  $F$ , enquanto que *args*() captura a lista de argumentos ( $A, B$ )



# Lógica de Primeira Ordem – Resolução

## Unificador mais Geral (UMG): Algoritmo

**Função**  $Unifica(x,y,s)$ : substituição

**se**  $s = falha$  **então retorna** *falha*

**se**  $x = y$  **então retorna**  $s$

**se**  $x$  *for uma variável* **então**

└ **retorna**  $UnificaVar(x,y,s)$

**se**  $y$  *for uma variável* **então**

└ **retorna**  $UnificaVar(y,x,s)$

**se**  $x$  e  $y$  *forem expressões compostas* **então**

└ **retorna**  $Unifica(args(x), args(y), Unifica(op(x), op(y), s))$

**se**  $x$  e  $y$  *forem listas* **então**

└ **retorna**  $Unifica(resto(x), resto(y), Unifica(primeiro(x), primeiro(y), s))$

└ **retorna** *falha*

Unificamos antes os operadores.  
E estes precisam ser idênticos

# Lógica de Primeira Ordem – Resolução

## Unificador mais Geral (UMG): Algoritmo

**Função** *Unifica*(*x*,*y*,*s*): substituição

**se** *s* = *falha* **então retorna** *falha*

**se** *x* = *y* **então retorna** *s*

**se** *x* *for uma variável* **então**

└ **retorna** *UnificaVar*(*x*,*y*,*s*)

**se** *y* *for uma variável* **então**

└ **retorna** *UnificaVar*(*y*,*x*,*s*)

**se** *x* e *y* *forem expressões compostas* **então**

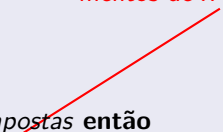
└ **retorna** *Unifica*(*args*(*x*), *args*(*y*), *Unifica*(*op*(*x*),*op*(*y*),*s*))

**se** *x* e *y* *forem listas* **então**

└ **retorna** *Unifica*(*resto*(*x*), *resto*(*y*), *Unifica*(*primeiro*(*x*),  
*primeiro*(*y*), *s*))

└ **retorna** *falha*

Unificamos então a lista de argumentos de *x* e *y* no contexto de *s*



# Lógica de Primeira Ordem – Resolução

## Unificador mais Geral (UMG): Algoritmo

**Função** *Unifica*(*x*,*y*,*s*): substituição

**se** *s* = *falha* **então retorna** *falha*

**se** *x* = *y* **então retorna** *s*

**se** *x* *for uma variável* **então**

└ **retorna** *UnificaVar*(*x*,*y*,*s*)

**se** *y* *for uma variável* **então**

└ **retorna** *UnificaVar*(*y*,*x*,*s*)

**se** *x* e *y* *forem expressões compostas* **então**

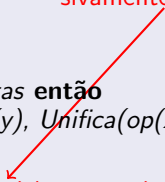
└ **retorna** *Unifica*(*args*(*x*), *args*(*y*), *Unifica*(*op*(*x*),*op*(*y*),*s*))

**se** *x* e *y* *forem listas* **então**

└ **retorna** *Unifica*(*resto*(*x*), *resto*(*y*), *Unifica*(*primeiro*(*x*),  
*primeiro*(*y*), *s*))

**retorna** *falha*

No caso de listas, tentamos unificar o primeiro elemento, para então recursivamente unificar os demais



# Lógica de Primeira Ordem – Resolução

## Unificador mais Geral (UMG): Algoritmo

**Função** *Unifica*(*x*,*y*,*s*): **substituição**

**se** *s* = *falha* **então retorna** *falha*

**se** *x* = *y* **então retorna** *s*

**se** *x* *for uma variável* **então**

└ **retorna** *UnificaVar*(*x*,*y*,*s*)

**se** *y* *for uma variável* **então**

└ **retorna** *UnificaVar*(*y*,*x*,*s*)

**se** *x* e *y* *forem expressões compostas* **então**

└ **retorna** *Unifica*(*args*(*x*), *args*(*y*), *Unifica*(*op*(*x*),*op*(*y*),*s*))

**se** *x* e *y* *forem listas* **então**

└ **retorna** *Unifica*(*resto*(*x*), *resto*(*y*), *Unifica*(*primeiro*(*x*),  
*primeiro*(*y*), *s*))

└ **retorna** *falha*

Nesse ponto, *x* e *y* são  
constantes distintas



## Unificador mais Geral (UMG): Algoritmo

**Função** *UnificaVar*(*var*, *x*, *s*): **substituição**

**se**  $\{var/val\} \in s$  **então**

└ **retorna** *Unifica*(*val*, *x*, *s*)

**se**  $\{x/val\} \in s$  **então**

└ **retorna** *Unifica*(*var*, *val*, *s*)

**se** *var* ocorre em *x*, com a substituição *s* aplicada a ele (*x s*)  
**então**

└ **retorna** falha (teste de ocorrência)

Adicione  $\{var/x\}$  a *s*

└ **retorna** *s*

## Unificador mais Geral (UMG): Algoritmo

**Função** *UnificaVar*(*var*, *x*, *s*): **substituição**

**se**  $\{var/val\} \in s$  **então**

└ **retorna** *Unifica*(*val*, *x*, *s*)

**se**  $\{x/val\} \in s$  **então**

└ **retorna** *Unifica*(*var*, *val*, *s*)

**se** *var* ocorre em *x*, com a substituição *s* aplicada a ele (*x s*)  
**então**

└ **retorna** falha (teste de ocorrência)

Adicione  $\{var/x\}$  a *s*

**retorna** *s*

Variável

# Lógica de Primeira Ordem – Resolução

## Unificador mais Geral (UMG): Algoritmo

**Função** *UnificaVar*(*var*, *x*, *s*): **substituição**

**se**  $\{var/val\} \in s$  **então**

└ **retorna** *Unifica*(*val*, *x*, *s*)

**se**  $\{x/val\} \in s$  **então**

└ **retorna** *Unifica*(*var*, *val*, *s*)

**se** *var* ocorre em *x*, com a substituição *s* aplicada a ele (*x s*)  
**então**

└ **retorna** falha (teste de ocorrência)

Adicione  $\{var/x\}$  a *s*

└ **retorna** *s*

Expressão

## Unificador mais Geral (UMG): Algoritmo

**Função** *UnificaVar*(*var*, *x*, *s*): **substituição**

**se**  $\{var/val\} \in s$  **então**

└ **retorna** *Unifica*(*val*, *x*, *s*)

Substituição atual

**se**  $\{x/val\} \in s$  **então**

└ **retorna** *Unifica*(*var*, *val*, *s*)

**se** *var* ocorre em *x*, com a substituição *s* aplicada a ele (*x s*)  
**então**

└ **retorna** falha (teste de ocorrência)

Adicione  $\{var/x\}$  a *s*

**retorna** *s*

# Lógica de Primeira Ordem – Resolução

## Unificador mais Geral (UMG): Algoritmo

**Função** *UnificaVar*(*var*, *x*, *s*): **substituição**

**se**  $\{var/val\} \in s$  **então**

└ **retorna** *Unifica*(*val*, *x*, *s*)

**se**  $\{x/val\} \in s$  **então**

└ **retorna** *Unifica*(*var*, *val*, *s*)

**se** *var* ocorre em *x*, com a substituição *s* aplicada a ele (*x s*)  
**então**

└ **retorna** falha (teste de ocorrência)

Adicione  $\{var/x\}$  a *s*

**retorna** *s*

Primeiro aplicamos as substituições existentes em *s* a *var* e *x*

# Lógica de Primeira Ordem – Resolução

## Unificador mais Geral (UMG): Algoritmo

**Função** *UnificaVar*(*var*, *x*, *s*): **substituição**

**se**  $\{var/val\} \in s$  **então**

└ **retorna** *Unifica*(*val*, *x*, *s*)

**se**  $\{x/val\} \in s$  **então**

└ **retorna** *Unifica*(*var*, *val*, *s*)

**se** *var* ocorre em *x*, com a substituição *s* aplicada a ele (*x s*)  
**então**

└ **retorna** falha (teste de ocorrência)

Adicione  $\{var/x\}$  a *s*

**retorna** *s*

*val* pode ser um valor ou expressão

# Lógica de Primeira Ordem – Resolução

## Unificador mais Geral (UMG): Algoritmo

**Função** *UnificaVar*(*var*, *x*, *s*): **substituição**

**se**  $\{var/val\} \in s$  **então**

└ **retorna** *Unifica*(*val*, *x*, *s*)

Unificamos então a substituição recém feita à outra expressão (*x* ou *var*)

**se**  $\{x/val\} \in s$  **então**

└ **retorna** *Unifica*(*var*, *val*, *s*)

**se** *var* ocorre em *x*, com a substituição *s* aplicada a ele (*x s*) **então**

└ **retorna** falha (teste de ocorrência)

Adicione  $\{var/x\}$  a *s*

└ **retorna** *s*

## Unificador mais Geral (UMG): Algoritmo

**Função** *UnificaVar*(*var*, *x*, *s*): **substituição**

**se**  $\{var/val\} \in s$  **então**

└ **retorna** *Unifica*(*val*, *x*, *s*)

**se**  $\{x/val\} \in s$  **então**

└ **retorna** *Unifica*(*var*, *val*, *s*)


**se** *var* **ocorre em** *x*, **com a substituição** *s* **aplicada a ele** (*x s*)  
**então**

└ **retorna** falha (teste de ocorrência)

Adicione  $\{var/x\}$  a *s*

**retorna** *s*

No início,  $s = \emptyset \rightarrow$

$$Ps = P, \forall P$$




## Unificador mais Geral (UMG): Algoritmo

**Função** *UnificaVar*(*var*, *x*, *s*): **substituição**

**se**  $\{var/val\} \in s$  **então**

└ **retorna** *Unifica*(*val*, *x*, *s*)

**se**  $\{x/val\} \in s$  **então**

└ **retorna** *Unifica*(*var*, *val*, *s*)

**se** *var* **ocorre em** *x*, **com a substituição** *s* **aplicada a ele** (*x s*)  
**então**

└ **retorna** falha (teste de ocorrência)

Adicione  $\{var/x\}$  a *s*

**retorna** *s*

Evita laços como  $\{x/f(x)\}$



# Lógica de Primeira Ordem – Resolução

## Unificador mais Geral (UMG): Algoritmo

**Função** *UnificaVar*(*var*, *x*, *s*): **substituição**

**se**  $\{var/val\} \in s$  **então**

└ **retorna** *Unifica*(*val*, *x*, *s*)

**se**  $\{x/val\} \in s$  **então**

└ **retorna** *Unifica*(*var*, *val*, *s*)

**se** *var* ocorre em *x*, com a substituição *s* aplicada a ele (*x s*)  
**então**

└ **retorna** falha (teste de ocorrência)

Adicione  $\{var/x\}$  a *s*

**retorna** *s*

Se nem *var* nem *x* participarem de *s*, e passarem no teste de ocorrência, então substituímos a variável *var* pelo que estiver em *x*, de modo a alinhá-las

# Lógica de Primeira Ordem – Resolução

## Unificador mais Geral (UMG): Algoritmo

**Função** *UnificaVar*(*var*, *x*, *s*): **substituição**

**se**  $\{var/val\} \in s$  **então**

└ **retorna** *Unifica*(*val*, *x*, *s*)

**se**  $\{x/val\} \in s$  **então**

└ **retorna** *Unifica*(*var*, *val*, *s*)

**se** *var* ocorre em *x*, com a substituição *s* aplicada a ele (*x s*)  
**então**

└ **retorna** falha (teste de ocorrência)

Adicione  $\{var/x\}$  a *s*

**retorna** *s*

Substituímos a variável  
justamente por ser mais  
flexível → por aceitar  
atribuições facilmente

# Referências

- Russell, S.; Norvig P. (2010): Artificial Intelligence: A Modern Approach. Prentice Hall. 3a ed.
  - Slides do livro: <http://aima.eecs.berkeley.edu/slides-pdf/>
- Hiž, A. (1957): Inferential Equivalence and Natural Deduction. The Journal of Symbolic Logic, 22(3). pp. 237-240.
- <http://ocw.mit.edu/OcwWeb/Electrical-Engineering-and-Computer-Science/6-034Spring-2005/LectureNotes/index.htm>
- <http://jmvidal.cse.sc.edu/talks/learningrules/first-orderlogicsdefs.xml>
- <https://www.sciencedirect.com/topics/computer-science/unification-algorithm>
- [http://logic.stanford.edu/intrologic/secondary/notes/chapter\\_12.html](http://logic.stanford.edu/intrologic/secondary/notes/chapter_12.html)