Inteligência Artificial – ACH2016
Aula 12 – Resolução e Forward Chaining em Lógica de Primeira Ordem

Norton Trevisan Roman
(norton@usp.br)

9 de abril de 2019

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

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

$$\Downarrow$$

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

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

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

$$\Downarrow$$

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

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

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

$$\Downarrow$$

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

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

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

$$\Downarrow$$

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

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

Lógica de Primeira Ordem – Resolução

Regra de inferência para resolução

$$\frac{\begin{array}{c}\alpha \lor \phi \\ \neg\psi \lor \beta\end{array}}{(\alpha \lor \beta)\theta} \qquad \mathit{UMG}(\phi, \psi) = \theta$$

Lógica de Primeira Ordem – Resolução

Regra de inferência para resolução

$$\frac{\begin{array}{c}\alpha \vee \phi \\ \neg\psi \vee \beta\end{array}}{(\alpha \vee \beta)\theta} \qquad \mathit{UMG}(\phi, \psi) = \theta$$

Como em lógica proposicional, unifica literais complementares

Lógica de Primeira Ordem – Resolução

Regra de inferência para resolução

$$\frac{\begin{array}{c}\alpha \vee \phi \\ \neg\psi \vee \beta\end{array}}{(\alpha \vee \beta)\theta} \qquad UMG(\phi, \psi) = \theta$$

Literais complementares em LPO: se um pode ser unificado à negação do outro

Lógica de Primeira Ordem – Resolução

Regra de inferência para resolução

$$\frac{\begin{array}{c}\alpha \vee \phi \\ \neg\psi \vee \beta\end{array}}{(\alpha \vee \beta)\theta} \qquad UMG(\phi, \psi) = \theta$$

Literais complementares em LPO: se um pode ser unificado à negação do outro

- Se tivermos $\alpha \vee \phi$ e $\neg\psi \vee \beta$, e pudermos unificar ϕ e ψ com θ, então podemos concluir $\alpha \vee \beta$ com a substituição θ aplicada a eles

Lógica de Primeira Ordem – Resolução

Regra de inferência para resolução

$$\frac{\begin{array}{c}\alpha \vee \phi \\ \neg\psi \vee \beta\end{array}}{(\alpha \vee \beta)\theta} \qquad UMG(\phi, \psi) = \theta$$

Literais complementares em LPO: se um pode ser unificado à negação do outro

- Se tivermos $\alpha \vee \phi$ e $\neg\psi \vee \beta$, e pudermos unificar ϕ e ψ com θ, então podemos concluir $\alpha \vee \beta$ com a substituição θ aplicada a eles
- Ambas cláusulas devem estar padronizadas

Lógica de Primeira Ordem – Resolução

Regra de inferência para resolução

$$\frac{\begin{array}{c}\alpha \vee \phi \\ \neg\psi \vee \beta\end{array}}{(\alpha \vee \beta)\theta} \qquad \mathit{UMG}(\phi, \psi) = \theta$$

Literais complementares em LPO: se um pode ser unificado à negação do outro

- Se tivermos $\alpha \vee \phi$ e $\neg\psi \vee \beta$, e pudermos unificar ϕ e ψ com θ, então podemos concluir $\alpha \vee \beta$ com a substituição θ aplicada a eles

- <u>Ambas cláusulas devem estar padronizadas</u>
  - Ou seja, sem variáveis em comum

Regra de inferência: Exemplo

$$\frac{\begin{array}{c}P(x) \vee Q(x, y) \\ \neg P(A) \vee R(B, x)\end{array}}{}$$

Regra de inferência: Exemplo

Lembre que há um quantificador universal implícito em cada cláusula

$$\frac{\begin{array}{c} P(x) \vee Q(x,y) \\ \neg P(A) \vee R(B,x) \end{array}}{} \quad \Rightarrow \quad \frac{\begin{array}{c} \forall x, y \; P(x) \vee Q(x,y) \\ \forall x \; \neg P(A) \vee R(B,x) \end{array}}{}$$

Lógica de Primeira Ordem – Resolução

Regra de inferência: Exemplo

Lembre que há um quantificador universal implícito em cada cláusula

$$\frac{\begin{array}{c} P(x) \vee Q(x, y) \\ \neg P(A) \vee R(B, x) \end{array}}{} \quad \Rightarrow \quad \frac{\begin{array}{c} \forall x, y \ P(x) \vee Q(x, y) \\ \forall x \ \neg P(A) \vee R(B, x) \end{array}}{}$$

Os x's nas duas cláusulas são diferentes

Regra de inferência: Exemplo

Lembre que há um quantificador universal implícito em cada cláusula

$$\begin{array}{c} P(x) \vee Q(x, y) \\ \neg P(A) \vee R(B, x) \\ \hline \end{array} \quad \Rightarrow \quad \begin{array}{c} \forall x, y \; P(x) \vee Q(x, y) \\ \forall x \; \neg P(A) \vee R(B, x) \\ \hline \end{array}$$

Os x's nas duas cláusulas são diferentes

Para evitar confusão, renomeamos as variáveis

$$\begin{array}{c} \forall x, y \; P(x) \vee Q(x, y) \\ \forall x_1 \; \neg P(A) \vee R(B, x_1) \\ \hline \end{array}$$

Regra de inferência: Exemplo

$$\frac{\begin{array}{c} P(x) \vee Q(x,y) \\ \neg P(A) \vee R(B,x) \end{array}}{} \quad \Rightarrow \quad \frac{\begin{array}{c} \forall x, y \; P(x) \vee Q(x,y) \\ \forall x \; \neg P(A) \vee R(B,x) \end{array}}{}$$

Lembre que há um quantificador universal implícito em cada cláusula

Os x's nas duas cláusulas são diferentes

Para evitar confusão, renomeamos as variáveis

$$\frac{\begin{array}{c} \forall x, y \; P(x) \vee Q(x,y) \\ \forall x_1 \; \neg P(A) \vee R(B,x_1) \end{array}}{} \quad \Rightarrow \quad \frac{\begin{array}{c} P(x) \vee Q(x,y) \\ \neg P(A) \vee R(B,x_1) \end{array}}{}$$

Lógica de Primeira Ordem – Resolução

Regra de inferência: Exemplo

$$\frac{\begin{array}{c} P(x) \lor Q(x, y) \\ \neg P(A) \lor R(B, x_1) \end{array}}{}$$

Lógica de Primeira Ordem – Resolução

Regra de inferência: Exemplo

$$\frac{\begin{array}{c} P(x) \vee Q(x,y) \\ \neg P(A) \vee R(B,x_1) \end{array}}{}$$

$$\frac{\begin{array}{c} \alpha \vee \phi \\ \neg\psi \vee \beta \end{array}}{(\alpha \vee \beta)\theta} \qquad UMG(\phi,\psi) = \theta$$

Lógica de Primeira Ordem – Resolução

Regra de inferência: Exemplo

$$\frac{\begin{array}{c} P(x) \lor Q(x,y) \\ \neg P(A) \lor R(B,x_1) \end{array}}{}$$

Buscamos dois literais
que neguem um o outro

$$\frac{\begin{array}{c} \alpha \lor \phi \\ \neg\psi \lor \beta \end{array}}{(\alpha \lor \beta)\theta} \qquad UMG(\phi,\psi) = \theta$$

Lógica de Primeira Ordem – Resolução

Regra de inferência: Exemplo

$$\frac{\begin{array}{c} P(x) \lor Q(x,y) \\ \neg P(A) \lor R(B,x_1) \end{array}}{}$$

E tentamos unificá-los

$$\frac{\begin{array}{c} \alpha \lor \phi \\ \neg\psi \lor \beta \end{array}}{(\alpha \lor \beta)\theta} \qquad UMG(\phi,\psi) = \theta$$

Lógica de Primeira Ordem – Resolução

Regra de inferência: Exemplo

$$\frac{\begin{array}{c} P(x) \vee Q(x,y) \\ \neg P(A) \vee R(B,x_1) \end{array}}{}$$

E tentamos unificá-los

$$\frac{\begin{array}{c} \alpha \vee \phi \\ \neg\psi \vee \beta \end{array}}{(\alpha \vee \beta)\theta} \qquad UMG(\phi,\psi) = \theta$$

$$\frac{\begin{array}{c} P(x) \vee Q(x,y) \\ \neg P(A) \vee R(B,x_1) \end{array}}{(Q(x,y) \vee R(B,x_1))\theta} \qquad \theta =$$

Lógica de Primeira Ordem – Resolução

Regra de inferência: Exemplo

$$\frac{\begin{array}{c} P(x) \vee Q(x,y) \\ \neg P(A) \vee R(B,x_1) \end{array}}{}$$

E tentamos unificá-los

$$\frac{\begin{array}{c} \alpha \vee \phi \\ \neg\psi \vee \beta \end{array}}{(\alpha \vee \beta)\theta} \qquad UMG(\phi,\psi) = \theta$$

$$\frac{\begin{array}{c} P(x) \vee Q(x,y) \\ \neg P(A) \vee R(B,x_1) \end{array}}{(Q(x,y) \vee R(B,x_1))\theta} \qquad \theta = \{x/A\}$$

Regra de inferência: Exemplo

$$\frac{\begin{array}{c} P(x) \vee Q(x,y) \\ \neg P(A) \vee R(B,x_1) \end{array}}{}$$

E tentamos unificá-los

$$\frac{\begin{array}{c} \alpha \vee \phi \\ \neg\psi \vee \beta \end{array}}{(\alpha \vee \beta)\theta} \qquad UMG(\phi,\psi) = \theta$$

$$\frac{\begin{array}{c} P(x) \vee Q(x,y) \\ \neg P(A) \vee R(B,x_1) \end{array}}{(Q(x,y) \vee R(B,x_1))\theta} \qquad \theta = \{x/A\} \quad \Rightarrow \quad \frac{\begin{array}{c} P(A) \vee Q(A,y) \\ \neg P(A) \vee R(B,x_1) \end{array}}{Q(A,y) \vee R(B,x_1)}$$

Lógica de Primeira Ordem – Resolução

Resolução: Exemplo

- Imagine que temos a seguinte base:

  João tem um cachorro

  Quem tem um cachorro é um amante dos animais

  Amantes dos animais não os matam

  João matou Bolão ou a curiosidade o fez

  Bolão é um gato

  Todo gato é um animal

Resolução: Exemplo

- Imagine que temos a seguinte base:

  João tem um cachorro

  Quem tem um cachorro é um amante dos animais

  Amantes dos animais não os matam

  João matou Bolão ou a curiosidade o fez

  Bolão é um gato

  Todo gato é um animal

- Queremos então saber:
  - A curiosidade matou Bolão?

Exemplo: Construindo a base

Exemplo: Construindo a base

João tem um cachorro
$\exists x\ Cão(x) \land Ter(João, x)$
$Cão(Totó) \land Ter(João, Totó)$

Exemplo: Construindo a base

João tem um cachorro
$\exists x\ Cão(x) \wedge Ter(João, x)$
$Cão(Totó) \wedge Ter(João, Totó)$

Quem tem um cachorro é um amante dos animais
$\forall x\ (\exists y\ Cão(y) \wedge Ter(x, y)) \Rightarrow AmaAnimais(x)$
$\forall x\ (\neg\exists y\ Cão(y) \wedge Ter(x, y)) \vee AmaAnimais(x)$
$\forall x \forall y\ \neg(Cão(y) \wedge Ter(x, y)) \vee AmaAnimais(x)$
$\forall x \forall y\ \neg Cão(y) \vee \neg Ter(x, y) \vee AmaAnimais(x)$
$\neg Cão(y) \vee \neg Ter(x, y) \vee AmaAnimais(x)$

Exemplo: Construindo a base

João tem um cachorro
$\exists x\ Cão(x) \wedge Ter(João, x)$
$Cão(Totó) \wedge Ter(João, Totó)$

Quem tem um cachorro é um amante dos animais
$\forall x\ (\exists y\ Cão(y) \wedge Ter(x, y)) \Rightarrow AmaAnimais(x)$
$\forall x\ (\neg\exists y\ Cão(y) \wedge Ter(x, y)) \vee AmaAnimais(x)$
$\forall x \forall y\ \neg(Cão(y) \wedge Ter(x, y)) \vee AmaAnimais(x)$
$\forall x \forall y\ \neg Cão(y) \vee \neg Ter(x, y) \vee AmaAnimais(x)$
$\neg Cão(y) \vee \neg Ter(x, y) \vee AmaAnimais(x)$

Amantes dos animais não os matam
$\forall x\ AmaAnimais(x) \Rightarrow (\forall y\ Animal(y) \Rightarrow \neg Matar(x, y))$
$\forall x\ \neg AmaAnimais(x) \vee (\forall y\ Animal(y) \Rightarrow \neg Matar(x, y))$
$\forall x\ \neg AmaAnimais(x) \vee (\forall y\ \neg Animal(y) \vee \neg Matar(x, y))$
$\neg AmaAnimais(x) \vee \neg Animal(y) \vee \neg Matar(x, y)$

Lógica de Primeira Ordem – Resolução

Exemplo: Construindo a base

João tem um cachorro
$\exists x\ Cão(x) \land Ter(João, x)$
$Cão(Totó) \land Ter(João, Totó)$

João matou Bolão ou a curiosidade o fez
$Matar(João, Bolão) \lor Matar(Curiosidade, Bolão)$

Quem tem um cachorro é um amante dos animais
$\forall x\ (\exists y\ Cão(y) \land Ter(x, y)) \Rightarrow AmaAnimais(x)$
$\forall x\ (\neg\exists y\ Cão(y) \land Ter(x, y)) \lor AmaAnimais(x)$
$\forall x \forall y\ \neg(Cão(y) \land Ter(x, y)) \lor AmaAnimais(x)$
$\forall x \forall y\ \neg Cão(y) \lor \neg Ter(x, y) \lor AmaAnimais(x)$
$\neg Cão(y) \lor \neg Ter(x, y) \lor AmaAnimais(x)$

Amantes dos animais não os matam
$\forall x\ AmaAnimais(x) \Rightarrow (\forall y\ Animal(y) \Rightarrow \neg Matar(x, y))$
$\forall x\ \neg AmaAnimais(x) \lor (\forall y\ Animal(y) \Rightarrow \neg Matar(x, y))$
$\forall x\ \neg AmaAnimais(x) \lor (\forall y\ \neg Animal(y) \lor \neg Matar(x, y))$
$\neg AmaAnimais(x) \lor \neg Animal(y) \lor \neg Matar(x, y)$

Exemplo: Construindo a base

João tem um cachorro
$\exists x\ Cão(x) \wedge Ter(João, x)$
$Cão(Totó) \wedge Ter(João, Totó)$

João matou Bolão ou a curiosidade o fez
$Matar(João, Bolão) \vee Matar(Curiosidade, Bolão)$

Quem tem um cachorro é um amante dos animais
$\forall x\ (\exists y\ Cão(y) \wedge Ter(x, y)) \Rightarrow AmaAnimais(x)$
$\forall x\ (\neg\exists y\ Cão(y) \wedge Ter(x, y)) \vee AmaAnimais(x)$
$\forall x \forall y\ \neg(Cão(y) \wedge Ter(x, y)) \vee AmaAnimais(x)$
$\forall x \forall y\ \neg Cão(y) \vee \neg Ter(x, y) \vee AmaAnimais(x)$
$\neg Cão(y) \vee \neg Ter(x, y) \vee AmaAnimais(x)$

Bolão é um gato
$Gato(Bolão)$

Amantes dos animais não os matam
$\forall x\ AmaAnimais(x) \Rightarrow (\forall y\ Animal(y) \Rightarrow \neg Matar(x, y))$
$\forall x\ \neg AmaAnimais(x) \vee (\forall y\ Animal(y) \Rightarrow \neg Matar(x, y))$
$\forall x\ \neg AmaAnimais(x) \vee (\forall y\ \neg Animal(y) \vee \neg Matar(x, y))$
$\neg AmaAnimais(x) \vee \neg Animal(y) \vee \neg Matar(x, y)$

Lógica de Primeira Ordem – Resolução

Exemplo: Construindo a base

João tem um cachorro
$\exists x\ Cão(x) \wedge Ter(João, x)$
$Cão(Totó) \wedge Ter(João, Totó)$

João matou Bolão ou a curiosidade o fez
$Matar(João, Bolão) \vee Matar(Curiosidade, Bolão)$

Quem tem um cachorro é um amante dos animais
$\forall x\ (\exists y\ Cão(y) \wedge Ter(x, y)) \Rightarrow AmaAnimais(x)$
$\forall x\ (\neg\exists y\ Cão(y) \wedge Ter(x, y)) \vee AmaAnimais(x)$
$\forall x \forall y\ \neg(Cão(y) \wedge Ter(x, y)) \vee AmaAnimais(x)$
$\forall x \forall y\ \neg Cão(y) \vee \neg Ter(x, y) \vee AmaAnimais(x)$
$\neg Cão(y) \vee \neg Ter(x, y) \vee AmaAnimais(x)$

Bolão é um gato
$Gato(Bolão)$

Amantes dos animais não os matam
$\forall x\ AmaAnimais(x) \Rightarrow (\forall y\ Animal(y) \Rightarrow \neg Matar(x, y))$
$\forall x\ \neg AmaAnimais(x) \vee (\forall y\ Animal(y) \Rightarrow \neg Matar(x, y))$
$\forall x\ \neg AmaAnimais(x) \vee (\forall y\ \neg Animal(y) \vee \neg Matar(x, y))$
$\neg AmaAnimais(x) \vee \neg Animal(y) \vee \neg Matar(x, y)$

Todo gato é um animal
$\forall x\ Gato(x) \Rightarrow Animal(x)$
$\forall x\ \neg Gato(x) \vee Animal(x)$
$\neg Gato(x) \vee Animal(x)$

Lógica de Primeira Ordem – Resolução

Exemplo: Inferência

	Sentença	*Ação*
1	$Cão(Totó)$	Axioma
2	$Ter(João, Totó)$	Axioma
3	$\neg Cão(y) \vee \neg Ter(x, y) \vee AmaAnimais(x)$	Axioma
4	$\neg AmaAnimais(x) \vee \neg Animal(y) \vee \neg Matar(x, y)$	Axioma
5	$Matar(João, Bolão) \vee Matar(Curiosidade, Bolão)$	Axioma
6	$Gato(Bolão)$	Axioma
7	$\neg Gato(x) \vee Animal(x)$	Axioma

Lógica de Primeira Ordem – Resolução

Exemplo: Inferência

	Sentença	*Ação*
1	$Cão(Totó)$	Axioma
2	$Ter(João, Totó)$	Axioma
3	$\neg Cão(y) \vee \neg Ter(x, y) \vee AmaAnimais(x)$	Axioma
4	$\neg AmaAnimais(x) \vee \neg Animal(y) \vee \neg Matar(x, y)$	Axioma
5	$Matar(João, Bolão) \vee Matar(Curiosidade, Bolão)$	Axioma
6	$Gato(Bolão)$	Axioma
7	$\neg Gato(x) \vee Animal(x)$	Axioma

- A curiosidade matou Bolão?

Lógica de Primeira Ordem – Resolução

Exemplo: Inferência

	Sentença	*Ação*
1	$Cão(Totó)$	Axioma
2	$Ter(João, Totó)$	Axioma
3	$\neg Cão(y) \vee \neg Ter(x, y) \vee AmaAnimais(x)$	Axioma
4	$\neg AmaAnimais(x) \vee \neg Animal(y) \vee \neg Matar(x, y)$	Axioma
5	$Matar(João, Bolão) \vee Matar(Curiosidade, Bolão)$	Axioma
6	$Gato(Bolão)$	Axioma
7	$\neg Gato(x) \vee Animal(x)$	Axioma

- A curiosidade matou Bolão?
  - Se queremos provar isso, incluímos sua negação na base (ou seja, assumimos o oposto disso)

Exemplo: Inferência

	Sentença	*Ação*
1	$Cão(Totó)$	Axioma
2	$Ter(João, Totó)$	Axioma
3	$\neg Cão(y) \vee \neg Ter(x, y) \vee AmaAnimais(x)$	Axioma
4	$\neg AmaAnimais(x) \vee \neg Animal(y) \vee \neg Matar(x, y)$	Axioma
5	$Matar(João, Bolão) \vee Matar(Curiosidade, Bolão)$	Axioma
6	$Gato(Bolão)$	Axioma
7	$\neg Gato(x) \vee Animal(x)$	Axioma
8	$\neg Matar(Curiosidade, Bolão)$	Negação

- A curiosidade matou Bolão?
  - Se queremos provar isso, incluímos sua negação na base (ou seja, assumimos o oposto disso)

Lógica de Primeira Ordem – Resolução

Exemplo: Inferência

	Sentença	*Ação*
1	$Cão(Totó)$	Axioma
2	$Ter(João, Totó)$	Axioma
3	$\neg Cão(y) \vee \neg Ter(x, y) \vee AmaAnimais(x)$	Axioma
4	$\neg AmaAnimais(x) \vee \neg Animal(y) \vee \neg Matar(x, y)$	Axioma
5	$Matar(João, Bolão) \vee Matar(Curiosidade, Bolão)$	Axioma
6	$Gato(Bolão)$	Axioma
7	$\neg Gato(x) \vee Animal(x)$	Axioma
8	$\neg Matar(Curiosidade, Bolão)$	Negação

- Podemos então aplicar a regra da resolução a quaisquer pares de linhas que contenham literais unificáveis

Lógica de Primeira Ordem – Resolução

Exemplo: Inferência

	Sentença	Ação
1	$Cão(Totó)$	Axioma
2	$Ter(João, Totó)$	Axioma
3	$\neg Cão(y) \lor \neg Ter(x, y) \lor AmaAnimais(x)$	Axioma
4	$\neg AmaAnimais(x) \lor \neg Animal(y) \lor \neg Matar(x, y)$	Axioma
5	$Matar(João, Bolão) \lor Matar(Curiosidade, Bolão)$	Axioma
6	$Gato(Bolão)$	Axioma
7	$\neg Gato(x) \lor Animal(x)$	Axioma
8	$\neg Matar(Curiosidade, Bolão)$	Negação

- Lembrando que a resolução prova que $BC \models \alpha$ através da prova de que $BC \land \neg\alpha$ é insatisfatível, ou seja, derivando a cláusula vazia (*falso*)

Lógica de Primeira Ordem – Resolução

Exemplo: Inferência

	Sentença	*Ação*
1	$Cão(Totó)$	Axioma
2	$Ter(João, Totó)$	Axioma
3	$\neg Cão(y) \vee \neg Ter(x, y) \vee AmaAnimais(x)$	Axioma
4	$\neg AmaAnimais(x) \vee \neg Animal(y) \vee \neg Matar(x, y)$	Axioma
5	$Matar(João, Bolão) \vee Matar(Curiosidade, Bolão)$	Axioma
6	$Gato(Bolão)$	Axioma
7	$\neg Gato(x) \vee Animal(x)$	Axioma
8	$\neg Matar(Curiosidade, Bolão)$	Negação

- Podemos começar usando a **Heurística do conjunto de suporte**: envolvemos a negação da conclusão na prova. Resolvemos então 5 e 8

Lógica de Primeira Ordem – Resolução

Exemplo: Inferência

	Sentença	Ação
1	$Cão(Totó)$	Axioma
2	$Ter(João, Totó)$	Axioma
3	$\neg Cão(y) \vee \neg Ter(x, y) \vee AmaAnimais(x)$	Axioma
4	$\neg AmaAnimais(x) \vee \neg Animal(y) \vee \neg Matar(x, y)$	Axioma
5	$Matar(João, Bolão) \vee Matar(Curiosidade, Bolão)$	Axioma
6	$Gato(Bolão)$	Axioma
7	$\neg Gato(x) \vee Animal(x)$	Axioma
8	$\neg Matar(Curiosidade, Bolão)$	Negação
9	$Matar(João, Bolão)$	5,8

- Podemos começar usando a **Heurística do conjunto de suporte**: envolvemos a negação da conclusão na prova. Resolvemos então 5 e 8

Exemplo: Inferência

	Sentença	Ação
1	$Cão(Totó)$	Axioma
2	$Ter(João, Totó)$	Axioma
3	$\neg Cão(y) \lor \neg Ter(x, y) \lor AmaAnimais(x)$	Axioma
4	$\neg AmaAnimais(x) \lor \neg Animal(y) \lor \neg Matar(x, y)$	Axioma
5	$Matar(João, Bolão) \lor Matar(Curiosidade, Bolão)$	Axioma
6	$Gato(Bolão)$	Axioma
7	$\neg Gato(x) \lor Animal(x)$	Axioma
8	$\neg Matar(Curiosidade, Bolão)$	Negação
9	$Matar(João, Bolão)$	5,8

Lógica de Primeira Ordem – Resolução

Exemplo: Inferência

	Sentença	*Ação*
1	$Cão(Totó)$	Axioma
2	$Ter(João, Totó)$	Axioma
3	$\neg Cão(y) \lor \neg Ter(x, y) \lor AmaAnimais(x)$	Axioma
4	$\neg AmaAnimais(x) \lor \neg Animal(y) \lor \neg Matar(x, y)$	Axioma
5	$Matar(João, Bolão) \lor Matar(Curiosidade, Bolão)$	Axioma
6	$Gato(Bolão)$	Axioma
7	$\neg Gato(x) \lor Animal(x)$	Axioma
8	$\neg Matar(Curiosidade, Bolão)$	Negação
9	$Matar(João, Bolão)$	5,8
10	$Animal(Bolão)$	6,7 $\{x/Bolão\}$

Exemplo: Inferência

	Sentença	*Ação*
1	$Cão(Totó)$	Axioma
2	$Ter(João, Totó)$	Axioma
3	$\neg Cão(y) \vee \neg Ter(x, y) \vee AmaAnimais(x)$	Axioma
4	$\neg AmaAnimais(x) \vee \neg Animal(y) \vee \neg Matar(x, y)$	Axioma
5	$Matar(João, Bolão) \vee Matar(Curiosidade, Bolão)$	Axioma
6	$Gato(Bolão)$	Axioma
7	$\neg Gato(x) \vee Animal(x)$	Axioma
8	$\neg Matar(Curiosidade, Bolão)$	Negação
9	$Matar(João, Bolão)$	5,8
10	$Animal(Bolão)$	6,7 $\{x/Bolão\}$
11	$\neg AmaAnimais(João) \vee \neg Animal(Bolão)$	4,9 $\{x/João, y/Bolão\}$

Lógica de Primeira Ordem – Resolução

Exemplo: Inferência

	Sentença	*Ação*
1	$Cão(Totó)$	Axioma
2	$Ter(João, Totó)$	Axioma
3	$\neg Cão(y) \vee \neg Ter(x, y) \vee AmaAnimais(x)$	Axioma
4	$\neg AmaAnimais(x) \vee \neg Animal(y) \vee \neg Matar(x, y)$	Axioma
5	$Matar(João, Bolão) \vee Matar(Curiosidade, Bolão)$	Axioma
6	$Gato(Bolão)$	Axioma
7	$\neg Gato(x) \vee Animal(x)$	Axioma
8	$\neg Matar(Curiosidade, Bolão)$	Negação
9	$Matar(João, Bolão)$	5,8
10	$Animal(Bolão)$	6,7 $\{x/Bolão\}$
11	$\neg AmaAnimais(João) \vee \neg Animal(Bolão)$	4,9 $\{x/João, y/Bolão\}$
12	$\neg AmaAnimais(João)$	10,11

Exemplo: Inferência

	Sentença	Ação
1	$Cão(Totó)$	Axioma
2	$Ter(João, Totó)$	Axioma
3	$\neg Cão(y) \lor \neg Ter(x, y) \lor AmaAnimais(x)$	Axioma
4	$\neg AmaAnimais(x) \lor \neg Animal(y) \lor \neg Matar(x, y)$	Axioma
5	$Matar(João, Bolão) \lor Matar(Curiosidade, Bolão)$	Axioma
6	$Gato(Bolão)$	Axioma
7	$\neg Gato(x) \lor Animal(x)$	Axioma
8	$\neg Matar(Curiosidade, Bolão)$	Negação
9	$Matar(João, Bolão)$	5,8
10	$Animal(Bolão)$	6,7 $\{x/Bolão\}$
11	$\neg AmaAnimais(João) \lor \neg Animal(Bolão)$	4,9 $\{x/João, y/Bolão\}$
12	$\neg AmaAnimais(João)$	10,11
13	$\neg Cão(y) \lor \neg Ter(João, y)$	3,12 $\{x/João\}$

Lógica de Primeira Ordem – Resolução

Exemplo: Inferência

	Sentença	*Ação*
1	$Cão(Totó)$	Axioma
2	$Ter(João, Totó)$	Axioma
3	$\neg Cão(y) \vee \neg Ter(x, y) \vee AmaAnimais(x)$	Axioma
4	$\neg AmaAnimais(x) \vee \neg Animal(y) \vee \neg Matar(x, y)$	Axioma
5	$Matar(João, Bolão) \vee Matar(Curiosidade, Bolão)$	Axioma
6	$Gato(Bolão)$	Axioma
7	$\neg Gato(x) \vee Animal(x)$	Axioma
8	$\neg Matar(Curiosidade, Bolão)$	Negação
9	$Matar(João, Bolão)$	5,8
10	$Animal(Bolão)$	6,7 $\{x/Bolão\}$
11	$\neg AmaAnimais(João) \vee \neg Animal(Bolão)$	4,9 $\{x/João, y/Bolão\}$
12	$\neg AmaAnimais(João)$	10,11
13	$\neg Cão(y) \vee \neg Ter(João, y)$	3,12 $\{x/João\}$
14	$\neg Cão(Totó)$	13,2 $\{y/Totó\}$

Exemplo: Inferência

	Sentença	Ação
1	$Cão(Totó)$	Axioma
2	$Ter(João, Totó)$	Axioma
3	$\neg Cão(y) \vee \neg Ter(x, y) \vee AmaAnimais(x)$	Axioma
4	$\neg AmaAnimais(x) \vee \neg Animal(y) \vee \neg Matar(x, y)$	Axioma
5	$Matar(João, Bolão) \vee Matar(Curiosidade, Bolão)$	Axioma
6	$Gato(Bolão)$	Axioma
7	$\neg Gato(x) \vee Animal(x)$	Axioma
8	$\neg Matar(Curiosidade, Bolão)$	Negação
9	$Matar(João, Bolão)$	5,8
10	$Animal(Bolão)$	6,7 $\{x/Bolão\}$
11	$\neg AmaAnimais(João) \vee \neg Animal(Bolão)$	4,9 $\{x/João, y/Bolão\}$
12	$\neg AmaAnimais(João)$	10,11
13	$\neg Cão(y) \vee \neg Ter(João, y)$	3,12 $\{x/João\}$
14	$\neg Cão(Totó)$	13,2 $\{y/Totó\}$
15	*falso*	1,14

Resolução Binária

Lógica de Primeira Ordem – Resolução

Resolução Binária

- É a regra da resolução vista até agora

Lógica de Primeira Ordem – Resolução

Resolução Binária

- É a regra da resolução vista até agora
  - Envolve apenas dois literais, um de cada cláusula resolvida

Resolução Binária

- É a regra da resolução vista até agora
  - Envolve apenas dois literais, um de cada cláusula resolvida
- Problema: não é completa

Resolução Binária

- É a regra da resolução vista até agora
  - Envolve apenas dois literais, um de cada cláusula resolvida
- Problema: não é completa
  - Há conjuntos de cláusulas insatisfatíveis (falsas em todas as interpretações) que não gerarão contradição através da aplicação sucessiva da regra

Resolução Binária

- É a regra da resolução vista até agora
  - Envolve apenas dois literais, um de cada cláusula resolvida
- Problema: não é completa
  - Há conjuntos de cláusulas insatisfatíveis (falsas em todas as interpretações) que não gerarão contradição através da aplicação sucessiva da regra
- Exemplo:

Lógica de Primeira Ordem – Resolução

Resolução Binária

- É a regra da resolução vista até agora
  - Envolve apenas dois literais, um de cada cláusula resolvida
- Problema: não é completa
  - Há conjuntos de cláusulas insatisfatíveis (falsas em todas as interpretações) que não gerarão contradição através da aplicação sucessiva da regra
- Exemplo:
  - Podemos obter uma contradição a partir dessas cláusulas?

    $P(x) \vee P(y)$

    $\neg P(v) \vee \neg P(w)$

Lógica de Primeira Ordem – Resolução

Resolução Binária

- É a regra da resolução vista até agora
  - Envolve apenas dois literais, um de cada cláusula resolvida
- Problema: não é completa
  - Há conjuntos de cláusulas insatisfatíveis (falsas em todas as interpretações) que não gerarão contradição através da aplicação sucessiva da regra
- Exemplo:
  - Podemos obter uma contradição a partir dessas cláusulas?

$P(x) \vee P(y)$ Base

$\neg P(v) \vee \neg P(w)$

Lógica de Primeira Ordem – Resolução

Resolução Binária

- É a regra da resolução vista até agora
  - Envolve apenas dois literais, um de cada cláusula resolvida
- Problema: não é completa
  - Há conjuntos de cláusulas insatisfatíveis (falsas em todas as interpretações) que não gerarão contradição através da aplicação sucessiva da regra
- Exemplo:
  - Podemos obter uma contradição a partir dessas cláusulas?

$P(x) \lor P(y)$

$\neg P(v) \lor \neg P(w) \longleftarrow$ Negação do que queremos provar $(P(v) \land P(w))$

Lógica de Primeira Ordem – Resolução

Resolução Binária

- Deveríamos, pois não há interpretação que faça ambas as cláusulas serem verdadeiras. Lembre que elas significam:

  $\forall x, y\ P(x) \vee P(y)$

  $\forall v, w\ \neg P(v) \vee \neg P(w)$

Lógica de Primeira Ordem – Resolução

Resolução Binária

- Deveríamos, pois não há interpretação que faça ambas as cláusulas serem verdadeiras. Lembre que elas significam:

  $\forall x, y \ P(x) \vee P(y)$

  $\forall v, w \ \neg P(v) \vee \neg P(w)$

- Então para ambas serem verdadeiras, seria necessário que $P(x)$ e $\neg P(x)$ fossem verdadeiros, para todo x

Lógica de Primeira Ordem – Resolução

Resolução Binária

- Deveríamos, pois não há interpretação que faça ambas as cláusulas serem verdadeiras. Lembre que elas significam:

  $\forall x, y \ P(x) \vee P(y)$

  $\forall v, w \ \neg P(v) \vee \neg P(w)$

- Então para ambas serem verdadeiras, seria necessário que $P(x)$ e $\neg P(x)$ fossem verdadeiros, para todo x
- Contudo, aplicando resolução binária, obtemos, por exemplo

  $P(x) \vee \neg P(w), \ \theta = \{y/v\}$

Lógica de Primeira Ordem – Resolução

Resolução Binária

- Deveríamos, pois não há interpretação que faça ambas as cláusulas serem verdadeiras. Lembre que elas significam:

  $\forall x, y \; P(x) \vee P(y)$

  $\forall v, w \; \neg P(v) \vee \neg P(w)$

- Então para ambas serem verdadeiras, seria necessário que $P(x)$ e $\neg P(x)$ fossem verdadeiros, para todo x
- Contudo, aplicando resolução binária, obtemos, por exemplo

  $P(x) \vee \neg P(w), \; \theta = \{y/v\}$

- E repetindo o procedimento com uma das orações originais, voltamos a elas novamente

Resolução Generalizada

- Há no entanto uma extensão à resolução binária que é completa: a **resolução generalizada**

Lógica de Primeira Ordem – Resolução

Resolução Generalizada

- Há no entanto uma extensão à resolução binária que é completa: a **resolução generalizada**
  - Nela, buscamos subconjuntos de literais em uma cláusula que possam ser unificados com a negação de um subconjunto de literais na outra cláusula

Lógica de Primeira Ordem – Resolução

Resolução Generalizada

- Há no entanto uma extensão à resolução binária que é completa: a **resolução generalizada**
  - Nela, buscamos subconjuntos de literais em uma cláusula que possam ser unificados com a negação de um subconjunto de literais na outra cláusula
- Exemplo:

Lógica de Primeira Ordem – Resolução

Resolução Generalizada

- Há no entanto uma extensão à resolução binária que é completa: a **resolução generalizada**
  - Nela, buscamos subconjuntos de literais em uma cláusula que possam ser unificados com a negação de um subconjunto de literais na outra cláusula
- Exemplo:
  - Em "$P(x) \vee P(y)$" e "$\neg P(v) \vee \neg P(w)$", cada literal (P) em uma cláusula pode ser unificado com sua negação na outra

Lógica de Primeira Ordem – Resolução

Resolução Generalizada

- Há no entanto uma extensão à resolução binária que é completa: a **resolução generalizada**
  - Nela, buscamos subconjuntos de literais em uma cláusula que possam ser unificados com a negação de um subconjunto de literais na outra cláusula
- Exemplo:
  - Em "$P(x) \vee P(y)$" e "$\neg P(v) \vee \neg P(w)$", cada literal (P) em uma cláusula pode ser unificado com sua negação na outra
  - Gerando assim a contradição necessária (*falso*) para a demonstração

Fatoração

- Uma alternativa mais simples é introduzir uma nova regra de inferência, a ser usada com a resolução binária:

Lógica de Primeira Ordem – Resolução

Fatoração

- Uma alternativa mais simples é introduzir uma nova regra de inferência, a ser usada com a resolução binária:

$$\frac{\alpha \vee \beta \vee \gamma}{(\alpha \vee \gamma)\,\theta} \quad \theta = UMG(\alpha, \beta)$$

Lógica de Primeira Ordem – Resolução

Fatoração

- Uma alternativa mais simples é introduzir uma nova regra de inferência, a ser usada com a resolução binária:

$$\frac{\alpha \vee \beta \vee \gamma}{(\alpha \vee \gamma)\,\theta} \quad \theta = UMG(\alpha, \beta)$$

  - Se pudermos unificar dois literais na mesma cláusula, então podemos eliminar um deles (não importa qual), e então aplicar o unificador à cláusula inteira

Lógica de Primeira Ordem – Resolução

Fatoração

- Uma alternativa mais simples é introduzir uma nova regra de inferência, a ser usada com a resolução binária:

$$\frac{\alpha \vee \beta \vee \gamma}{(\alpha \vee \gamma)\,\theta} \quad \theta = \mathit{UMG}(\alpha, \beta)$$

  - Se pudermos unificar dois literais na mesma cláusula, então podemos eliminar um deles (não importa qual), e então aplicar o unificador à cláusula inteira

- Essa alternativa é chamada de **Fatoração**

Fatoração: Exemplo

- $$\frac{Q(y) \vee P(x,y) \vee P(v,A)}{}$$

Lógica de Primeira Ordem – Resolução

Fatoração: Exemplo

- $$\frac{Q(y) \vee P(x, y) \vee P(v, A)}{Q(A) \vee P(v, A)\ \{x/v, y/A\}}$$

Lógica de Primeira Ordem – Resolução

Fatoração: Exemplo

- $$\frac{Q(y) \vee P(x, y) \vee P(v, A)}{Q(A) \vee P(v, A)\ \{x/v, y/A\}}$$
  - Note que se trata da remoção de literais duplicados

Fatoração: Exemplo

- $$\frac{Q(y) \vee P(x, y) \vee P(v, A)}{Q(A) \vee P(v, A)\ \{x/v, y/A\}}$$
  - Note que se trata da remoção de literais duplicados
- Agora a resolução binária pode ser usada

Lógica de Primeira Ordem – Resolução

Fatoração: Exemplo

- $$\frac{Q(y) \vee P(x,y) \vee P(v,A)}{Q(A) \vee P(v,A)\ \{x/v, y/A\}}$$
  - Note que se trata da remoção de literais duplicados
- Agora a resolução binária pode ser usada
- A combinação de resolução binária com fatoração é completa

Fatoração: Exemplo

- $$\frac{Q(y) \vee P(x,y) \vee P(v,A)}{Q(A) \vee P(v,A)\ \{x/v, y/A\}}$$
  - Note que se trata da remoção de literais duplicados
- Agora a resolução binária pode ser usada
- A combinação de resolução binária com fatoração é completa
  - Toda vez que a base levar a S, poderemos provar S a partir dela

Respondendo Perguntas

- Até agora respondemos questões do tipo:
  - "A curiosidade matou Bolão?"

Respondendo Perguntas

- Até agora respondemos questões do tipo:
  - "A curiosidade matou Bolão?"
- Resolução pode, contudo, ser usada para responder perguntas mais gerais:

Respondendo Perguntas

- Até agora respondemos questões do tipo:
  - "A curiosidade matou Bolão?"
- Resolução pode, contudo, ser usada para responder perguntas mais gerais:
  - "Quem matou Bolão?"

Lógica de Primeira Ordem – Resolução

Respondendo Perguntas

- Até agora respondemos questões do tipo:
  - "A curiosidade matou Bolão?"
- Resolução pode, contudo, ser usada para responder perguntas mais gerais:
  - "Quem matou Bolão?"
  - "$Matar(w, Bolão)$"? (Olhando a substituição $\{w/???\}$)

Lógica de Primeira Ordem – Resolução

Respondendo Perguntas

- Até agora respondemos questões do tipo:
  - "A curiosidade matou Bolão?"
- Resolução pode, contudo, ser usada para responder perguntas mais gerais:
  - "Quem matou Bolão?"
  - "$Matar(w, Bolão)$"? (Olhando a substituição $\{w/???\}$)
- Como?

Lógica de Primeira Ordem – Resolução

Respondendo Perguntas

- Até agora respondemos questões do tipo:
  - "A curiosidade matou Bolão?"
- Resolução pode, contudo, ser usada para responder perguntas mais gerais:
  - "Quem matou Bolão?"
  - "$Matar(w, Bolão)$"? (Olhando a substituição $\{w/???\}$)
- Como? **Truque de (Cordell) Green**:

Lógica de Primeira Ordem – Resolução

Respondendo Perguntas

- Até agora respondemos questões do tipo:
  - "A curiosidade matou Bolão?"
- Resolução pode, contudo, ser usada para responder perguntas mais gerais:
  - "Quem matou Bolão?"
  - "$Matar(w, Bolão)$"? (Olhando a substituição $\{w/???\}$)
- Como? **Truque de (Cordell) Green**:
  - Adicione à negação do objetivo um **literal de resposta**

Lógica de Primeira Ordem – Resolução

Respondendo Perguntas

- Até agora respondemos questões do tipo:
  - "A curiosidade matou Bolão?"
- Resolução pode, contudo, ser usada para responder perguntas mais gerais:
  - "Quem matou Bolão?"
  - "$Matar(w, Bolão)$"? (Olhando a substituição $\{w/???\}$)
- Como? **Truque de (Cordell) Green**:
  - Adicione à negação do objetivo um **literal de resposta**
  - Ex: $\neg Matar(w, Bolão) \vee Resposta(w)$

Respondendo Perguntas: Exemplo

- Quem é mortal?

	Sentença	*Ação*
1	$\neg Homem(x) \vee Mortal(x)$	Axioma
2	$Homem(Sócrates)$	Axioma

Respondendo Perguntas: Exemplo

- Quem é mortal?

1. Negamos a conclusão (de que existem mortais)

	Sentença	*Ação*
1	$\neg Homem(x) \vee Mortal(x)$	Axioma
2	$Homem(Sócrates)$	Axioma
3	$\neg Mortal(x) \vee Resposta(x)$	Neg

Respondendo Perguntas: Exemplo

- Quem é mortal?

1. Negamos a conclusão (de que existem mortais)

	Sentença	*Ação*
1	$\neg Homem(x) \vee Mortal(x)$	Axioma
2	$Homem(Sócrates)$	Axioma
3	$\neg Mortal(x) \vee Resposta(x)$	Neg

2. O truque é adicionar o literal Resposta(x)

Respondendo Perguntas: Exemplo

- Quem é mortal?

1. Negamos a conclusão (de que existem mortais)

2. O truque é adicionar o literal Resposta(x)

3. Fazemos a resolução

	Sentença	*Ação*
1	$\neg Homem(x) \vee Mortal(x)$	Axioma
2	$Homem(Sócrates)$	Axioma
3	$\neg Mortal(x) \vee Resposta(x)$	Neg
4	$Mortal(Sócrates)$	1,2

Respondendo Perguntas: Exemplo

- Quem é mortal?

	Sentença	*Ação*
1	$\neg Homem(x) \vee Mortal(x)$	Axioma
2	$Homem(Sócrates)$	Axioma
3	$\neg Mortal(x) \vee Resposta(x)$	Neg
4	$Mortal(Sócrates)$	1,2
5	$Resposta(Sócrates)$	3,5

1. Negamos a conclusão (de que existem mortais)

2. O truque é adicionar o literal Resposta(x)

3. Fazemos a resolução

4. Quando chegamos a uma cláusula com apenas o literal Resposta, paramos

Lógica de Primeira Ordem – Resolução

Respondendo Perguntas: Exemplo

- Quem é mortal?

	Sentença	Ação
1	$\neg Homem(x) \lor Mortal(x)$	Axioma
2	$Homem(Sócrates)$	Axioma
3	$\neg Mortal(x) \lor Resposta(x)$	Neg
4	$Mortal(Sócrates)$	1,2
5	$Resposta(Sócrates)$	3,5

1. Negamos a conclusão (de que existem mortais)

2. O truque é adicionar o literal Resposta(x)

3. Fazemos a resolução

4. Quando chegamos a uma cláusula com apenas o literal Resposta, paramos

5. A resposta será o que estiver ligado à variável x nesse literal

Lógica de Primeira Ordem – Resolução

Igualdade

- Nenhum dos métodos de inferência até agora lidam com $x = y$

Lógica de Primeira Ordem – Resolução

Igualdade

- Nenhum dos métodos de inferência até agora lidam com $x = y$
- O que fazer quando a prova contém um '$=$'?

Lógica de Primeira Ordem – Resolução

Igualdade

- Nenhum dos métodos de inferência até agora lidam com $x = y$
- O que fazer quando a prova contém um '$=$'?
  - Adicione uma regra extra – **paramodulação** (não veremos)

Lógica de Primeira Ordem – Resolução

Igualdade

- Nenhum dos métodos de inferência até agora lidam com $x = y$
- O que fazer quando a prova contém um '$=$'?
  - Adicione uma regra extra – **paramodulação** (não veremos)
    - Usada em provadores de teoremas

Lógica de Primeira Ordem – Resolução

Igualdade

- Nenhum dos métodos de inferência até agora lidam com $x = y$
- O que fazer quando a prova contém um '$=$'?
  - Adicione uma regra extra – **paramodulação** (não veremos)
    - Usada em provadores de teoremas
  - Estenda o algoritmo de unificação (não veremos)

Lógica de Primeira Ordem – Resolução

Igualdade

- Nenhum dos métodos de inferência até agora lidam com $x = y$
- O que fazer quando a prova contém um '$=$'?
  - Adicione uma regra extra – **paramodulação** (não veremos)
    - Usada em provadores de teoremas
  - Estenda o algoritmo de unificação (não veremos)
    - Usada em provadores de teoremas

Lógica de Primeira Ordem – Resolução

Igualdade

- Nenhum dos métodos de inferência até agora lidam com $x = y$
- O que fazer quando a prova contém um '$=$'?
  - Adicione uma regra extra – **paramodulação** (não veremos)
    - Usada em provadores de teoremas
  - Estenda o algoritmo de unificação (não veremos)
    - Usada em provadores de teoremas
  - Trate a igualdade como qualquer outro predicado

Igualdade

- Nenhum dos métodos de inferência até agora lidam com $x = y$
- O que fazer quando a prova contém um '$=$'?
  - Adicione uma regra extra – **paramodulação** (não veremos)
    - Usada em provadores de teoremas
  - Estenda o algoritmo de unificação (não veremos)
    - Usada em provadores de teoremas
  - Trate a igualdade como qualquer outro predicado
    - Restrinja sua semântica via axiomas

Lógica de Primeira Ordem – Resolução

Axiomatizando Igualdade

Lógica de Primeira Ordem – Resolução

Axiomatizando Igualdade

- Aproximamos a igualdade por equivalência, tomando por base suas propriedades

Lógica de Primeira Ordem – Resolução

Axiomatizando Igualdade

- Aproximamos a igualdade por equivalência, tomando por base suas propriedades
  - $\forall x \; Igual(x, x)$ (reflexividade)

Lógica de Primeira Ordem – Resolução

Axiomatizando Igualdade

- Aproximamos a igualdade por equivalência, tomando por base suas propriedades
  - $\forall x\ Igual(x, x)$ (reflexividade)
  - $\forall x, y\ Igual(x, y) \Rightarrow Igual(y, x)$ (simetria)

Axiomatizando Igualdade

- Aproximamos a igualdade por equivalência, tomando por base suas propriedades
  - $\forall x\ Igual(x, x)$ (reflexividade)
  - $\forall x, y\ Igual(x, y) \Rightarrow Igual(y, x)$ (simetria)
  - $\forall x, y, z\ Igual(x, y) \wedge Igual(y, z) \Rightarrow Igual(x, z)$ (transitividade)

Lógica de Primeira Ordem – Resolução

Axiomatizando Igualdade

- Aproximamos a igualdade por equivalência, tomando por base suas propriedades
  - $\forall x\ Igual(x, x)$ (reflexividade)
  - $\forall x, y\ Igual(x, y) \Rightarrow Igual(y, x)$ (simetria)
  - $\forall x, y, z\ Igual(x, y) \wedge Igual(y, z) \Rightarrow Igual(x, z)$ (transitividade)
- Permitimos então substituição, em qualquer predicado ou função, de termos iguais:

Lógica de Primeira Ordem – Resolução

Axiomatizando Igualdade

- Aproximamos a igualdade por equivalência, tomando por base suas propriedades
  - $\forall x \; Igual(x, x)$ (reflexividade)
  - $\forall x, y \; Igual(x, y) \Rightarrow Igual(y, x)$ (simetria)
  - $\forall x, y, z \; Igual(x, y) \wedge Igual(y, z) \Rightarrow Igual(x, z)$ (transitividade)
- Permitimos então substituição, em qualquer predicado ou função, de termos iguais:
  - $\forall x, y \; Igual(x, y) \Rightarrow (P_i(x) \Leftrightarrow P_i(y))$, para todo P_i

Lógica de Primeira Ordem – Resolução

Axiomatizando Igualdade

- Aproximamos a igualdade por equivalência, tomando por base suas propriedades
  - $\forall x\ Igual(x, x)$ (reflexividade)
  - $\forall x, y\ Igual(x, y) \Rightarrow Igual(y, x)$ (simetria)
  - $\forall x, y, z\ Igual(x, y) \wedge Igual(y, z) \Rightarrow Igual(x, z)$ (transitividade)
- Permitimos então substituição, em qualquer predicado ou função, de termos iguais:
  - $\forall x, y\ Igual(x, y) \Rightarrow (P_i(x) \Leftrightarrow P_i(y))$, para todo P_i
  - Para cada P_i (e função) precisamos incluir um axioma assim

Forward Chaining

L.P.O. – Forward Chaining

Características

- Como na lógica proposicional:

L.P.O. – Forward Chaining

Características

- Como na lógica proposicional:
  - Funciona apenas com cláusulas definidas

L.P.O. – Forward Chaining

Características

- Como na lógica proposicional:
  - Funciona apenas com cláusulas definidas
    - $Antecedentes \Rightarrow Consequente$

L.P.O. – Forward Chaining

Características

- Como na lógica proposicional:
  - Funciona apenas com cláusulas definidas
    - $Antecedentes \Rightarrow Consequente$
  - A partir de sentenças atômicas, aplica Modus Ponens progressivamente, adicionando novas sentenças atômicas

L.P.O. – Forward Chaining

Características

- Como na lógica proposicional:
  - Funciona apenas com cláusulas definidas
    - $Antecedentes \Rightarrow Consequente$
  - A partir de sentenças atômicas, aplica Modus Ponens progressivamente, adicionando novas sentenças atômicas
- Utilidade

L.P.O. – Forward Chaining

Características

- Como na lógica proposicional:
  - Funciona apenas com cláusulas definidas
    - *Antecedentes* $\Rightarrow$ *Consequente*
  - A partir de sentenças atômicas, aplica Modus Ponens progressivamente, adicionando novas sentenças atômicas
- Utilidade
  - Sistemas que fazem inferência em resposta a informações recém-chegadas

L.P.O. – Forward Chaining

Características

- Como na lógica proposicional:
  - Funciona apenas com cláusulas definidas
    - *Antecedentes* $\Rightarrow$ *Consequente*
  - A partir de sentenças atômicas, aplica Modus Ponens progressivamente, adicionando novas sentenças atômicas
- Utilidade
  - Sistemas que fazem inferência em resposta a informações recém-chegadas
    - Sistemas baseados em regras, do tipo *Situação* $\Rightarrow$ *Resposta*

L.P.O. – Forward Chaining

Características

- Como na lógica proposicional:
  - Funciona apenas com cláusulas definidas
    - *Antecedentes* $\Rightarrow$ *Consequente*
  - A partir de sentenças atômicas, aplica Modus Ponens progressivamente, adicionando novas sentenças atômicas
- Utilidade
  - Sistemas que fazem inferência em resposta a informações recém-chegadas
    - Sistemas baseados em regras, do tipo *Situação* $\Rightarrow$ *Resposta*
  - Mais eficiente que resolução, nesses casos

L.P.O. – Forward Chaining

Relembrando Cláusulas Definidas

- Disjunções de literais, dos quais exatamente um é positivo

L.P.O. – Forward Chaining

Relembrando Cláusulas Definidas

- Disjunções de literais, dos quais exatamente um é positivo
  - $\neg P(x) \vee \neg Q(x) \vee \neg R(x) \vee K(x)$, ou

L.P.O. – Forward Chaining

Relembrando Cláusulas Definidas

- Disjunções de literais, dos quais exatamente um é positivo
  - $\neg P(x) \vee \neg Q(x) \vee \neg R(x) \vee K(x)$, ou
  - $P(x) \wedge Q(x) \wedge R(x) \Rightarrow K(x)$

    (Conjunção de literais positivos $\Rightarrow$ único literal positivo)

L.P.O. – Forward Chaining

Relembrando Cláusulas Definidas

- Disjunções de literais, dos quais exatamente um é positivo
  - $\neg P(x) \vee \neg Q(x) \vee \neg R(x) \vee K(x)$, ou
  - $P(x) \wedge Q(x) \wedge R(x) \Rightarrow K(x)$

    (Conjunção de literais positivos $\Rightarrow$ único literal positivo)
- Ou uma sentença atômica

L.P.O. – Forward Chaining

Relembrando Cláusulas Definidas

- Disjunções de literais, dos quais exatamente um é positivo
  - $\neg P(x) \vee \neg Q(x) \vee \neg R(x) \vee K(x)$, ou
  - $P(x) \wedge Q(x) \wedge R(x) \Rightarrow K(x)$

    (Conjunção de literais positivos $\Rightarrow$ único literal positivo)
- Ou uma sentença atômica
  - $P(x)$

L.P.O. – Forward Chaining

Relembrando Cláusulas Definidas

- Disjunções de literais, dos quais exatamente um é positivo
  - $\neg P(x) \vee \neg Q(x) \vee \neg R(x) \vee K(x)$, ou
  - $P(x) \wedge Q(x) \wedge R(x) \Rightarrow K(x)$

    (Conjunção de literais positivos $\Rightarrow$ único literal positivo)
- Ou uma sentença atômica
  - $P(x)$
  - $Gato(Bolão)$

L.P.O. – Forward Chaining

Bases de Conhecimento com Cláusulas Definidas

L.P.O. – Forward Chaining

Bases de Conhecimento com Cláusulas Definidas

- Cada entrada na base é uma cláusula definida

L.P.O. – Forward Chaining

Bases de Conhecimento com Cláusulas Definidas

- Cada entrada na base é uma cláusula definida
  - $P(x)$

L.P.O. – Forward Chaining

Bases de Conhecimento com Cláusulas Definidas

- Cada entrada na base é uma cláusula definida
  - $P(x)$
  - $P(x) \land Q(x) \land R(x) \Rightarrow K(x)$

L.P.O. – Forward Chaining

Bases de Conhecimento com Cláusulas Definidas

- Cada entrada na base é uma cláusula definida
  - $P(x)$
  - $P(x) \wedge Q(x) \wedge R(x) \Rightarrow K(x)$
  - Traz implícita, em cada entrada, o quantificador universal

L.P.O. – Forward Chaining

Bases de Conhecimento com Cláusulas Definidas

- Cada entrada na base é uma cláusula definida
  - $\forall x \; P(x)$
  - $\forall x \; P(x) \land Q(x) \land R(x) \Rightarrow K(x)$
  - Traz implícita, em cada entrada, o quantificador universal

L.P.O. – Forward Chaining

Bases de Conhecimento com Cláusulas Definidas

- Cada entrada na base é uma cláusula definida
  - $\forall x\ P(x)$
  - $\forall x\ P(x) \wedge Q(x) \wedge R(x) \Rightarrow K(x)$
  - Traz implícita, em cada entrada, o quantificador universal
- Também aqui precisamos eliminar o quantificador existencial (skolemização)

L.P.O. – Forward Chaining

Bases de Conhecimento com Cláusulas Definidas

- Cada entrada na base é uma cláusula definida
  - $\forall x \; P(x)$
  - $\forall x \; P(x) \land Q(x) \land R(x) \Rightarrow K(x)$
  - Traz implícita, em cada entrada, o quantificador universal
- Também aqui precisamos eliminar o quantificador existencial (skolemização)
- Além de padronizar as variáveis das entradas que serão unificadas (como na Resolução)

L.P.O. – Forward Chaining

Funcionamento

L.P.O. – Forward Chaining

Funcionamento

- Começando com os fatos conhecidos, verifique todas as regras cujas premissas são satisfeitas

L.P.O. – Forward Chaining

Funcionamento

- Começando com os fatos conhecidos, verifique todas as regras cujas premissas são satisfeitas
  - Modus Ponens

$$\frac{\alpha \Rightarrow \beta \quad \psi}{(\beta)\,\theta} \quad UMG(\alpha, \psi) = \theta$$

$$\frac{P(x) \Rightarrow Q(x) \quad P(A)}{Q(A)\,\{x/A\}}$$

L.P.O. – Forward Chaining

Funcionamento

- Começando com os fatos conhecidos, verifique todas as regras cujas premissas são satisfeitas
  - Modus Ponens

$$\frac{\begin{array}{c}\alpha \Rightarrow \beta \\ \psi\end{array}}{(\beta)\,\theta} \quad UMG(\alpha, \psi) = \theta \qquad\qquad \frac{\begin{array}{c}P(x) \Rightarrow Q(x) \\ P(A)\end{array}}{Q(A)\,\{x/A\}}$$

- Adicione suas conclusões aos fatos conhecidos

L.P.O. – Forward Chaining

Funcionamento

- Começando com os fatos conhecidos, verifique todas as regras cujas premissas são satisfeitas
  - Modus Ponens

$$\frac{\alpha \Rightarrow \beta \qquad \psi}{(\beta)\,\theta} \quad UMG(\alpha,\psi) = \theta \qquad\qquad \frac{P(x) \Rightarrow Q(x) \qquad P(A)}{Q(A)\,\{x/A\}}$$

- Adicione suas conclusões aos fatos conhecidos
- Repita o procedimento até que a query seja respondida (assumindo que uma resposta apenas baste), ou nenhum novo fato puder ser adicionado

L.P.O. – Forward Chaining

Funcionamento

- Um fato não será novo se for apenas uma renomeação de um fato conhecido

L.P.O. – Forward Chaining

Funcionamento

- Um fato não será novo se for apenas uma renomeação de um fato conhecido
  - Ou seja, se ambos significados forem idênticos

L.P.O. – Forward Chaining

Funcionamento

- Um fato não será novo se for apenas uma renomeação de um fato conhecido
  - Ou seja, se ambos significados forem idênticos
- Uma sentença é uma renomeação de outra se forem idênticas, exceto pelos nomes de suas variáveis

L.P.O. – Forward Chaining

Funcionamento

- Um fato não será novo se for apenas uma renomeação de um fato conhecido
  - Ou seja, se ambos significados forem idênticos
- Uma sentença é uma renomeação de outra se forem idênticas, exceto pelos nomes de suas variáveis
  - Ex: $Gosta(x, Sorvete)$ e $Gosta(y, Sorvete)$

L.P.O. – Forward Chaining

Exemplo

- Base:
  - É um crime para um compatriota vender armas a nações hostis. Pitbulândia, um inimigo nosso, tem alguns mísseis, e todos seus mísseis foram vendidos a eles por um compatriota: Osmar

L.P.O. – Forward Chaining

Exemplo

- Base:
  - É um crime para um compatriota vender armas a nações hostis. Pitbulândia, um inimigo nosso, tem alguns mísseis, e todos seus mísseis foram vendidos a eles por um compatriota: Osmar
- Osmar é criminoso?

L.P.O. – Forward Chaining

Exemplo

- Base:
  - É um crime para um compatriota vender armas a nações hostis. Pitbulândia, um inimigo nosso, tem alguns mísseis, e todos seus mísseis foram vendidos a eles por um compatriota: Osmar
- Osmar é criminoso?

1. $\forall x, y, z \ Compatriota(x) \wedge Vender(x, y, z) \wedge Arma(y) \wedge Hostil(z) \Rightarrow Criminoso(x)$

$$Compatriota(x) \wedge Vender(x, y, z) \wedge Arma(y) \wedge Hostil(z) \Rightarrow Criminoso(x)$$

L.P.O. – Forward Chaining

Exemplo

- Base:
  - É um crime para um compatriota vender armas a nações hostis. Pitbulândia, um inimigo nosso, tem alguns mísseis, e todos seus mísseis foram vendidos a eles por um compatriota: Osmar

- Osmar é criminoso?

1. $\forall x, y, z \; Compatriota(x) \land Vender(x,y,z) \land Arma(y) \land Hostil(z) \Rightarrow Criminoso(x)$
   $Compatriota(x) \land Vender(x,y,z) \land Arma(y) \land Hostil(z) \Rightarrow Criminoso(x)$

2. $Inimigo(Pit)$

L.P.O. – Forward Chaining

Exemplo

- Base:
  - É um crime para um compatriota vender armas a nações hostis. Pitbulândia, um inimigo nosso, tem alguns mísseis, e todos seus mísseis foram vendidos a eles por um compatriota: Osmar
- Osmar é criminoso?

1. $\forall x, y, z \; Compatriota(x) \wedge Vender(x, y, z) \wedge Arma(y) \wedge Hostil(z) \Rightarrow Criminoso(x)$
$Compatriota(x) \wedge Vender(x, y, z) \wedge Arma(y) \wedge Hostil(z) \Rightarrow Criminoso(x)$

2. $Inimigo(Pit)$

3. $\exists x \; Míssil(x) \wedge Ter(Pit, x)$
$Míssil(M) \wedge Ter(Pit, M)$

L.P.O. – Forward Chaining

Exemplo

- Base:
  - É um crime para um compatriota vender armas a nações hostis. Pitbulândia, um inimigo nosso, tem alguns mísseis, e todos seus mísseis foram vendidos a eles por um compatriota: Osmar
- Osmar é criminoso?

$$\boxed{\begin{array}{c} 4.\ \forall x\ \mathit{Míssil}(x) \wedge \mathit{Ter}(\mathit{Pit}, x) \Rightarrow \mathit{Vender}(\mathit{Os}, x, \mathit{Pit}) \\ \mathit{Míssil}(x) \wedge \mathit{Ter}(\mathit{Pit}, x) \Rightarrow \mathit{Vender}(\mathit{Os}, x, \mathit{Pit}) \end{array}}$$

L.P.O. – Forward Chaining

Exemplo

- Base:
  - É um crime para um compatriota vender armas a nações hostis. Pitbulândia, um inimigo nosso, tem alguns mísseis, e todos seus mísseis foram vendidos a eles por um compatriota: Osmar
- Osmar é criminoso?

$$\boxed{\begin{array}{c} 4.\ \forall x\ \mathit{Míssil}(x) \wedge Ter(Pit, x) \Rightarrow Vender(Os, x, Pit) \\ \mathit{Míssil}(x) \wedge Ter(Pit, x) \Rightarrow Vender(Os, x, Pit) \end{array}}$$

$$\boxed{\begin{array}{c} \text{5. Mísseis são armas} \\ \forall x\ \mathit{Míssil}(x) \Rightarrow Arma(x) \\ \mathit{Míssil}(x) \Rightarrow Arma(x) \end{array}}$$

L.P.O. – Forward Chaining

Exemplo

- Base:
  - É um crime para um compatriota vender armas a nações hostis. Pitbulândia, um inimigo nosso, tem alguns mísseis, e todos seus mísseis foram vendidos a eles por um compatriota: Osmar
- Osmar é criminoso?

$$\boxed{\begin{array}{c} 4.\ \forall x\ \mathit{Míssil}(x) \land \mathit{Ter}(\mathit{Pit}, x) \Rightarrow \mathit{Vender}(\mathit{Os}, x, \mathit{Pit}) \\ \mathit{Míssil}(x) \land \mathit{Ter}(\mathit{Pit}, x) \Rightarrow \mathit{Vender}(\mathit{Os}, x, \mathit{Pit}) \end{array}}$$

$$\boxed{\begin{array}{c} \text{5. Mísseis são armas} \\ \forall x\ \mathit{Míssil}(x) \Rightarrow \mathit{Arma}(x) \\ \mathit{Míssil}(x) \Rightarrow \mathit{Arma}(x) \end{array}} \longleftarrow$$

Note que essa sentença não está no texto → faz parte do **conhecimento comum**

L.P.O. – Forward Chaining

Exemplo

- Base:
  - É um crime para um compatriota vender armas a nações hostis. Pitbulândia, um inimigo nosso, tem alguns mísseis, e todos seus mísseis foram vendidos a eles por um compatriota: Osmar
- Osmar é criminoso?

6. Inimigos são hostis
$\forall x\ Inimigo(x) \Rightarrow Hostil(x)$
$Inimigo(x) \Rightarrow Hostil(x)$

L.P.O. – Forward Chaining

Exemplo

- Base:
  - É um crime para um compatriota vender armas a nações hostis. Pitbulândia, um inimigo nosso, tem alguns mísseis, e todos seus mísseis foram vendidos a eles por um compatriota: Osmar
- Osmar é criminoso?

6. Inimigos são hostis
$\forall x\ Inimigo(x) \Rightarrow Hostil(x)$
$Inimigo(x) \Rightarrow Hostil(x)$

$\longleftarrow$ Mais conhecimento comum ...

L.P.O. – Forward Chaining

Exemplo

- Base:
  - É um crime para um compatriota vender armas a nações hostis. Pitbulândia, um inimigo nosso, tem alguns mísseis, e todos seus mísseis foram vendidos a eles por um compatriota: Osmar
- Osmar é criminoso?

6. Inimigos são hostis
$\forall x \; Inimigo(x) \Rightarrow Hostil(x)$
$Inimigo(x) \Rightarrow Hostil(x)$

⟵ Mais conhecimento comum ...

$\boxed{7. \; Compatriota(Os)}$

L.P.O. – Forward Chaining

Exemplo

- Osmar é criminoso?

	Sentença	*Obs*
1	$Compatriota(x) \wedge Vender(x, y, z) \wedge Arma(y) \wedge Hostil(z) \Rightarrow Criminoso(x)$	axioma
2	$Inimigo(Pit)$	axioma
3	$Míssil(M)$	axioma
4	$Ter(Pit, M)$	axioma
5	$Míssil(x) \wedge Ter(Pit, x) \Rightarrow Vender(Os, x, Pit)$	axioma
6	$Míssil(x) \Rightarrow Arma(x)$	axioma
7	$Inimigo(x) \Rightarrow Hostil(x)$	axioma
8	$Compatriota(Os)$	axioma

L.P.O. – Forward Chaining

Exemplo

- Osmar é criminoso?

	Sentença	*Obs*
1	$Compatriota(x) \wedge Vender(x, y, z) \wedge Arma(y) \wedge Hostil(z) \Rightarrow Criminoso(x)$	axioma
2	$Inimigo(Pit)$	axioma
3	$Míssil(M)$	axioma
4	$Ter(Pit, M)$	axioma
5	$Míssil(x) \wedge Ter(Pit, x) \Rightarrow Vender(Os, x, Pit)$	axioma
6	$Míssil(x) \Rightarrow Arma(x)$	axioma
7	$Inimigo(x) \Rightarrow Hostil(x)$	axioma
8	$Compatriota(Os)$	axioma
9	$Arma(M)$	3,6 $\{x/M\}$

L.P.O. – Forward Chaining

Exemplo

- Osmar é criminoso?

	Sentença	*Obs*
1	$Compatriota(x) \wedge Vender(x, y, z) \wedge Arma(y) \wedge Hostil(z) \Rightarrow Criminoso(x)$	axioma
2	$Inimigo(Pit)$	axioma
3	$Míssil(M)$	axioma
4	$Ter(Pit, M)$	axioma
5	$Míssil(x) \wedge Ter(Pit, x) \Rightarrow Vender(Os, x, Pit)$	axioma
6	$Míssil(x) \Rightarrow Arma(x)$	axioma
7	$Inimigo(x) \Rightarrow Hostil(x)$	axioma
8	$Compatriota(Os)$	axioma
9	$Arma(M)$	3,6 $\{x/M\}$
10	$Hostil(Pit)$	2,7 $\{x/Pit\}$

L.P.O. – Forward Chaining

Exemplo

- Osmar é criminoso?

	Sentença	*Obs*
1	$Compatriota(x) \wedge Vender(x, y, z) \wedge$ $Arma(y) \wedge Hostil(z) \Rightarrow Criminoso(x)$	axioma
2	$Inimigo(Pit)$	axioma
3	$Míssil(M)$	axioma
4	$Ter(Pit, M)$	axioma
5	$Míssil(x) \wedge Ter(Pit, x) \Rightarrow Vender(Os, x, Pit)$	axioma
6	$Míssil(x) \Rightarrow Arma(x)$	axioma
7	$Inimigo(x) \Rightarrow Hostil(x)$	axioma
8	$Compatriota(Os)$	axioma
9	$Arma(M)$	3,6 $\{x/M\}$
10	$Hostil(Pit)$	2,7 $\{x/Pit\}$
11	$Vender(Os, M, Pit)$	3,4,5 $\{x/M\}$

L.P.O. – Forward Chaining

Exemplo

- Osmar é criminoso?

	Sentença	*Obs*
1	$Compatriota(x) \wedge Vender(x, y, z) \wedge Arma(y) \wedge Hostil(z) \Rightarrow Criminoso(x)$	axioma
2	$Inimigo(Pit)$	axioma
3	$Míssil(M)$	axioma
4	$Ter(Pit, M)$	axioma
5	$Míssil(x) \wedge Ter(Pit, x) \Rightarrow Vender(Os, x, Pit)$	axioma
6	$Míssil(x) \Rightarrow Arma(x)$	axioma
7	$Inimigo(x) \Rightarrow Hostil(x)$	axioma
8	$Compatriota(Os)$	axioma
9	$Arma(M)$	3,6 $\{x/M\}$
10	$Hostil(Pit)$	2,7 $\{x/Pit\}$
11	$Vender(Os, M, Pit)$	3,4,5 $\{x/M\}$
12	$Criminoso(Os)$	1,8,9,10,11 $\{x/Os, y/M, z/Pit\}$

Algoritmo

```
Função Forward-Chaining(BC,α): substituição
  repita
    novo ← {}
    para cada sentença s em BC faça
      (p1 ∧ ... ∧ pn ⇒ q) ← PadronizaVar(s)
      para cada θ tal que (p1 ∧ ... ∧ pn) θ = (p'1 ∧ ... ∧ p'n) θ para
        algum p'1, ..., p'n em BC faça
        q' ← (q) θ
        se q' não se unificar a alguma sentença em BC ou novo então
          Adicione q' a novo
          φ ← Unifica(q', α)
          se φ não for falha então  retorna φ
    Adicione novo a BC
  até que novo esteja vazio
  retorna falso
```

L.P.O. – Forward Chaining

Algoritmo

Função *Forward-Chaining(BC,*α*)*: **substituição**

repita

novo $\leftarrow \{\}$ Base de Conhecimento

para cada *sentença s em BC* **faça**

$(p_1 \wedge \ldots \wedge p_n \Rightarrow q) \leftarrow$ *PadronizaVar(s)*

para cada θ *tal que* $(p_1 \wedge \ldots \wedge p_n)\,\theta = (p'_1 \wedge \ldots \wedge p'_n)\,\theta$ *para algum* $p'_1, \ldots, p'_n$ *em BC* **faça**

$q' \leftarrow (q)\,\theta$

se q' *não se unificar a alguma sentença em BC ou novo* **então**

Adicione q' a *novo*

$\phi \leftarrow$ *Unifica*(q', α)

se ϕ *não for falha* **então** **retorna** ϕ

Adicione *novo* a *BC*

até *que novo esteja vazio*

retorna *falso*

L.P.O. – Forward Chaining

Algoritmo

Função *Forward-Chaining(BC,α)*: **substituição**

Query (sentença atômica)

```
repita
    novo ← {}
    para cada sentença s em BC faça
        (p₁ ∧ ... ∧ pₙ ⇒ q) ← PadronizaVar(s)
        para cada θ tal que (p₁ ∧ ... ∧ pₙ) θ = (p'₁ ∧ ... ∧ p'ₙ) θ para
          algum p'₁, ..., p'ₙ em BC faça
            q' ← (q) θ
            se q' não se unificar a alguma sentença em BC ou novo então
                Adicione q' a novo
                φ ← Unifica(q', α)
                se φ não for falha então  retorna φ
    Adicione novo a BC
até que novo esteja vazio
retorna falso
```

L.P.O. – Forward Chaining

Algoritmo

Função *Forward-Chaining(BC,α)*: **substituição**

repita

novo $\leftarrow \{\}$

para cada *sentença s em BC* **faça**

$(p_1 \wedge \ldots \wedge p_n \Rightarrow q) \leftarrow$ *PadronizaVar(s)*

para cada θ *tal que* $(p_1 \wedge \ldots \wedge p_n)\,\theta = (p'_1 \wedge \ldots \wedge p'_n)\,\theta$ *para algum* $p'_1, \ldots, p'_n$ *em BC* **faça**

$q' \leftarrow (q)\,\theta$

se q' *não se unificar a alguma sentença em BC ou novo* **então**

Adicione q' a *novo*

$\phi \leftarrow Unifica(q', \alpha)$

se ϕ *não for falha* **então** **retorna** ϕ

Adicione *novo* a *BC*

até *que novo esteja vazio*

retorna *falso*

Conjunto de novas sentenças inferidas a cada iteração

L.P.O. – Forward Chaining

Algoritmo

Função *Forward-Chaining(BC,α)*: **substituição**

repita

$novo \leftarrow \{\}$

para cada *sentença s em BC* **faça**

$(p_1 \wedge \ldots \wedge p_n \Rightarrow q) \leftarrow$ *PadronizaVar(s)*

para cada θ *tal que* $(p_1 \wedge \ldots \wedge p_n)\,\theta = (p'_1 \wedge \ldots \wedge p'_n)\,\theta$ *para algum* $p'_1, \ldots, p'_n$ *em BC* **faça**

$q' \leftarrow (q)\,\theta$

se q' *não se unificar a alguma sentença em BC ou novo* **então**

Adicione q' a *novo*

$\phi \leftarrow Unifica(q', \alpha)$

se ϕ *não for falha* **então** **retorna** ϕ

Adicione *novo* a *BC*

até *que novo esteja vazio*

retorna *falso*

PadronizaVar troca todas as variáveis em seus argumentos por variáveis ainda não usadas

L.P.O. – Forward Chaining

Algoritmo

Função *Forward-Chaining(BC,α)*: **substituição**

Começando dos fatos conhecidos, ativa todas as regras cujas premissas são satisfeitas

- **repita**
  - *novo* $\leftarrow \{\}$
  - **para cada** *sentença s em BC* **faça**
    - $(p_1 \wedge \ldots \wedge p_n \Rightarrow q) \leftarrow$ *PadronizaVar(s)*
    - **para cada** θ *tal que* $(p_1 \wedge \ldots \wedge p_n)\,\theta = (p'_1 \wedge \ldots \wedge p'_n)\,\theta$ *para algum* $p'_1, \ldots, p'_n$ *em BC* **faça**
      - $q' \leftarrow (q)\,\theta$
      - **se** q' *não se unificar a alguma sentença em BC ou novo* **então**
        - Adicione q' a *novo*
        - $\phi \leftarrow Unifica(q', \alpha)$
        - **se** ϕ *não for falha* **então** **retorna** ϕ
  - Adicione *novo* a *BC*
- **até** *que novo esteja vazio*
- **retorna** *falso*

L.P.O. – Forward Chaining

Algoritmo

Função *Forward-Chaining(BC,α)*: **substituição**

Adicionando sua conclusão aos fatos conhecidos

repita
  novo $\leftarrow \{\}$
  para cada *sentença s em BC* **faça**
    $(p_1 \wedge \ldots \wedge p_n \Rightarrow q) \leftarrow$ *PadronizaVar(s)*
    para cada θ *tal que* $(p_1 \wedge \ldots \wedge p_n)\,\theta = (p'_1 \wedge \ldots \wedge p'_n)\,\theta$ *para algum* $p'_1, \ldots, p'_n$ *em BC* **faça**
      $q' \leftarrow (q)\,\theta$
      se q' *não se unificar a alguma sentença em BC ou novo* **então**
        Adicione q' a *novo*
        $\phi \leftarrow$ *Unifica*(q', α)
        se ϕ *não for falha* **então** **retorna** ϕ
  Adicione *novo* a *BC*
até *que novo esteja vazio*
retorna *falso*

L.P.O. – Forward Chaining

Algoritmo

Função *Forward-Chaining(BC,α)*: **substituição**

repita

  novo $\leftarrow \{\}$

  para cada *sentença s em BC* **faça**

   $(p_1 \wedge \ldots \wedge p_n \Rightarrow q) \leftarrow$ *PadronizaVar(s)*

   para cada θ *tal que* $(p_1 \wedge \ldots \wedge p_n)\,\theta = (p'_1 \wedge \ldots \wedge p'_n)\,\theta$ *para algum* $p'_1, \ldots, p'_n$ *em BC* **faça**

    $q' \leftarrow (q)\,\theta$

    se *q' não se unificar a alguma sentença em BC ou novo* **então**

     Adicione q' a *novo*

     $\phi \leftarrow Unifica(q', \alpha)$

     se ϕ *não for falha* **então** **retorna** ϕ

  Adicione *novo* a *BC*

até *que novo esteja vazio*

retorna *falso*

Mas somente se o fato for desconhecido → se não é apenas uma renomeação de um conhecido

L.P.O. – Forward Chaining

Algoritmo

Função *Forward-Chaining(BC,α)*: **substituição**

repita

novo $\leftarrow \{\}$

para cada *sentença s em BC* **faça**

$(p_1 \wedge \ldots \wedge p_n \Rightarrow q) \leftarrow$ *PadronizaVar(s)*

para cada θ *tal que* $(p_1 \wedge \ldots \wedge p_n)\,\theta = (p'_1 \wedge \ldots \wedge p'_n)\,\theta$ *para algum* $p'_1, \ldots, p'_n$ *em BC* **faça**

$q' \leftarrow (q)\,\theta$

se q' *não se unificar a alguma sentença em BC ou novo* **então**

Adicione q' a *novo*

$\phi \leftarrow Unifica(q', \alpha)$

se ϕ *não for falha* **então** **retorna** ϕ

Adicione *novo* a *BC*

até *que novo esteja vazio*

retorna *falso*

Uma sentença é uma **renomeação** de outra se forem idênticas exceto pelo nome das variáveis

L.P.O. – Forward Chaining

Algoritmo

Função *Forward-Chaining(BC,α)*: **substituição**

O processo repete até que a query seja respondida ou nenhum fato extra possa ser adicionado

- **repita**
  - *novo* $\leftarrow \{\}$
  - **para cada** *sentença s em BC* **faça**
    - $(p_1 \wedge \ldots \wedge p_n \Rightarrow q) \leftarrow$ *PadronizaVar(s)*
    - **para cada** θ *tal que* $(p_1 \wedge \ldots \wedge p_n)\,\theta = (p'_1 \wedge \ldots \wedge p'_n)\,\theta$ *para algum* $p'_1, \ldots, p'_n$ *em BC* **faça**
      - $q' \leftarrow (q)\,\theta$
      - **se** q' *não se unificar a alguma sentença em BC ou novo* **então**
        - Adicione q' a *novo*
        - $\phi \leftarrow$ *Unifica*(q', α)
        - **se** ϕ *não for falha* **então** **retorna** ϕ
  - Adicione *novo* a *BC*
- **até** *que novo esteja vazio*
- **retorna** *falso*

Algoritmo: Problemas

L.P.O. – Forward Chaining

Algoritmo: Problemas

- Busca todos os unificadores possíveis θ

L.P.O. – Forward Chaining

Algoritmo: Problemas

- Busca todos os unificadores possíveis θ
  - É problema se para cada variável houver várias constantes que possam ser substituídas

L.P.O. – Forward Chaining

Algoritmo: Problemas

- Busca todos os unificadores possíveis θ
  - É problema se para cada variável houver várias constantes que possam ser substituídas
  - Solução: Olhe primeiro a variável que tem menos substituições possíveis (variável com mais restrições – PSR)

L.P.O. – Forward Chaining

Algoritmo: Problemas

- Busca todos os unificadores possíveis θ
  - É problema se para cada variável houver várias constantes que possam ser substituídas
  - Solução: Olhe primeiro a variável que tem menos substituições possíveis (variável com mais restrições – PSR)
- A cada iteração, verifica cada regra novamente

L.P.O. – Forward Chaining

Algoritmo: Problemas

- Busca todos os unificadores possíveis θ
  - É problema se para cada variável houver várias constantes que possam ser substituídas
  - Solução: Olhe primeiro a variável que tem menos substituições possíveis (variável com mais restrições – PSR)
- A cada iteração, verifica cada regra novamente
  - Mesmo que poucas adições tenham sido feitas à base em cada iteração

L.P.O. – Forward Chaining

Algoritmo: Problemas

- Busca todos os unificadores possíveis θ
  - É problema se para cada variável houver várias constantes que possam ser substituídas
  - Solução: Olhe primeiro a variável que tem menos substituições possíveis (variável com mais restrições – PSR)
- A cada iteração, verifica cada regra novamente
  - Mesmo que poucas adições tenham sido feitas à base em cada iteração
  - Todo fato novo inferido na iteração t deveria ser derivado de pelo menos um fato novo inferido na $t - 1$

L.P.O. – Forward Chaining

Algoritmo: Problemas

- Isso porque toda inferência que não necessita de um fato novo da iteração $t-1$ poderia ter sido feita em iterações anteriores

L.P.O. – Forward Chaining

Algoritmo: Problemas

- Isso porque toda inferência que não necessita de um fato novo da iteração $t-1$ poderia ter sido feita em iterações anteriores
- Solução: **forward chaining incremental**

L.P.O. – Forward Chaining

Algoritmo: Problemas

- Isso porque toda inferência que não necessita de um fato novo da iteração $t-1$ poderia ter sido feita em iterações anteriores
- Solução: **forward chaining incremental**
  - Na iteração t, verificamos uma regra somente se sua premissa incluir um p_i que se unifica com um fato p'_i recém inferido na iteração $t-1$

L.P.O. – Forward Chaining

Algoritmo: Problemas

- Isso porque toda inferência que não necessita de um fato novo da iteração $t - 1$ poderia ter sido feita em iterações anteriores
- Solução: **forward chaining incremental**
  - Na iteração t, verificamos uma regra somente se sua premissa incluir um p_i que se unifica com um fato p'_i recém inferido na iteração $t - 1$
  - Ou seja, case toda regra cuja premissa contiver um literal recém adicionado, permitindo, no entanto, que os demais componentes casem com fatos de iterações anteriores

L.P.O. – Forward Chaining

Algoritmo: Problemas

- Isso porque toda inferência que não necessita de um fato novo da iteração $t - 1$ poderia ter sido feita em iterações anteriores
- Solução: **forward chaining incremental**
    - Na iteração t, verificamos uma regra somente se sua premissa incluir um p_i que se unifica com um fato p'_i recém inferido na iteração $t - 1$
    - Ou seja, case toda regra cuja premissa contiver um literal recém adicionado, permitindo, no entanto, que os demais componentes casem com fatos de iterações anteriores
- Pode gerar muitos fatos irrelevantes ao objetivo

L.P.O. – Forward Chaining

Algoritmo: Problemas

- Isso porque toda inferência que não necessita de um fato novo da iteração $t-1$ poderia ter sido feita em iterações anteriores
- Solução: **forward chaining incremental**
  - Na iteração t, verificamos uma regra somente se sua premissa incluir um p_i que se unifica com um fato p'_i recém inferido na iteração $t-1$
  - Ou seja, case toda regra cuja premissa contiver um literal recém adicionado, permitindo, no entanto, que os demais componentes casem com fatos de iterações anteriores

- Pode gerar muitos fatos irrelevantes ao objetivo
  - Solução: Use backward chaining

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/`
- Hiż, 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`