Inteligência Artificial – ACH2016
Aula 09 – Inferência em Lógica Proposicional

Norton Trevisan Roman
(norton@usp.br)

10 de abril de 2019

Provas: Tipos

Provas: Tipos

- **Verificação do modelo**:

Provas: Tipos

- **Verificação do modelo**:
  - Enumeração de entradas em uma tabela-verdade

Provas: Tipos

- **Verificação do modelo**:
  - Enumeração de entradas em uma tabela-verdade
  - Força bruta

Provas: Tipos

- **Verificação do modelo**:
  - Enumeração de entradas em uma tabela-verdade
  - Força bruta
  - Algoritmos para CSPs resolvem

Provas: Tipos

- **Verificação do modelo**:
  - Enumeração de entradas em uma tabela-verdade
  - Força bruta
  - Algoritmos para CSPs resolvem
  - A que vimos até agora

Lógica Proposicional e Inferência

Provas: Tipos

- **Verificação do modelo**:
  - Enumeração de entradas em uma tabela-verdade
  - Força bruta
  - Algoritmos para CSPs resolvem
  - A que vimos até agora
- **Aplicação de regras de inferência**:

Lógica Proposicional e Inferência

Provas: Tipos

- **Verificação do modelo**:
  - Enumeração de entradas em uma tabela-verdade
  - Força bruta
  - Algoritmos para CSPs resolvem
  - A que vimos até agora
- **Aplicação de regras de inferência**:
  - Geração de novas sentenças a partir de antigas

Provas: Tipos

- **Verificação do modelo**:
  - Enumeração de entradas em uma tabela-verdade
  - Força bruta
  - Algoritmos para CSPs resolvem
  - A que vimos até agora
- **Aplicação de regras de inferência**:
  - Geração de novas sentenças a partir de antigas
  - A prova é uma sequência de regras de inferência

Provas: Tipos

- **Verificação do modelo**:
  - Enumeração de entradas em uma tabela-verdade
  - Força bruta
  - Algoritmos para CSPs resolvem
  - A que vimos até agora
- **Aplicação de regras de inferência**:
  - Geração de novas sentenças a partir de antigas
  - A prova é uma sequência de regras de inferência
  - Dedução natural

Provas: Aplicação de regras de inferência

Lógica Proposicional e Inferência

Provas: Aplicação de regras de inferência

- Trata da derivação de novas sentenças pela aplicação das leis da lógica proposicional

Provas: Aplicação de regras de inferência

- Trata da derivação de novas sentenças pela aplicação das leis da lógica proposicional
- São então sequências de sentenças

Lógica Proposicional e Inferência

Provas: Aplicação de regras de inferência

- Trata da derivação de novas sentenças pela aplicação das leis da lógica proposicional
- São então sequências de sentenças
  - Primeiro, listamos as premissas (as sentenças na base de conhecimento)

Lógica Proposicional e Inferência

Provas: Aplicação de regras de inferência

- Trata da derivação de novas sentenças pela aplicação das leis da lógica proposicional
- São então sequências de sentenças
  - Primeiro, listamos as premissas (as sentenças na base de conhecimento)
  - Em seguida, adicionamos resultados da aplicação das regras de inferência às sentenças conhecidas

Lógica Proposicional e Inferência

Provas: Aplicação de regras de inferência

- Trata da derivação de novas sentenças pela aplicação das leis da lógica proposicional
- São então sequências de sentenças
  - Primeiro, listamos as premissas (as sentenças na base de conhecimento)
  - Em seguida, adicionamos resultados da aplicação das regras de inferência às sentenças conhecidas
  - Quando chegamos à sentença Q (conclusão) desejada, achamos uma prova que leva da base à sentença

Provas: Aplicação de regras de inferência

- E se não chegarmos a Q?

Provas: Aplicação de regras de inferência

- E se não chegarmos a Q?
  - Então saberemos que $BC \models Q$ é falso

Provas: Aplicação de regras de inferência

- E se não chegarmos a Q?
  - Então saberemos que $BC \models Q$ é falso
- Mas não sabemos nada de Q

Provas: Aplicação de regras de inferência

- E se não chegarmos a Q?
  - Então saberemos que $BC \models Q$ é falso
- Mas não sabemos nada de Q
  - Sabemos que da base não deduzimos Q

Provas: Aplicação de regras de inferência

- E se não chegarmos a Q?
  - Então saberemos que $BC \models Q$ é falso
- Mas não sabemos nada de Q
  - Sabemos que da base não deduzimos Q
- Exemplo:

Provas: Aplicação de regras de inferência

- E se não chegarmos a Q?
  - Então saberemos que $BC \models Q$ é falso
- Mas não sabemos nada de Q
  - Sabemos que da base não deduzimos Q
- Exemplo:
  - Dado todo seu conhecimento sobre a sala de aula, está caindo um meteorito em marte?

Provas: Aplicação de regras de inferência

- E se não chegarmos a Q?
  - Então saberemos que $BC \models Q$ é falso
- Mas não sabemos nada de Q
  - Sabemos que da base não deduzimos Q
- Exemplo:
  - Dado todo seu conhecimento sobre a sala de aula, está caindo um meteorito em marte?
  - $BC \models Q = Falso$

Lógica Proposicional e Inferência

Regras de inferência

- **Modus Ponens**
  - (modo que afirma)
  - Proposto por Aristóteles

$$\frac{\begin{array}{c}\alpha \rightarrow \beta \\ \alpha\end{array}}{\beta}$$

Lógica Proposicional e Inferência

Regras de inferência

- **Modus Ponens**
  - (modo que afirma)
  - Proposto por Aristóteles

Premissas

$$\frac{\begin{array}{c}\alpha \rightarrow \beta \\ \alpha\end{array}}{\beta}$$

Regras de inferência

- **Modus Ponens**
  - (modo que afirma)
  - Proposto por Aristóteles

Conclusão

$$\frac{\begin{array}{c}\alpha \rightarrow \beta \\ \alpha\end{array}}{\beta}$$

Regras de inferência

- **Modus Ponens**
  - (modo que afirma)
  - Proposto por Aristóteles

$$\frac{\begin{array}{c}\alpha \rightarrow \beta \\ \alpha\end{array}}{\beta}$$

- **Modus Tollens**
  - (modo que nega)

$$\frac{\begin{array}{c}\alpha \rightarrow \beta \\ \neg\beta\end{array}}{\neg\alpha}$$

Regras de inferência

- **Introdução de E**

$$\frac{\begin{array}{c}\alpha \\ \beta\end{array}}{\alpha \land \beta}$$

Lógica Proposicional e Inferência

Regras de inferência

- **Introdução de E**

$$\frac{\begin{array}{c}\alpha \\ \beta\end{array}}{\alpha \wedge \beta}$$

- **Eliminação de E**

$$\frac{\alpha \wedge \beta}{\alpha} \qquad \frac{\alpha \wedge \beta}{\beta}$$

Regras de inferência: Exemplo

- Prove S

Passo	*Fórmula*	*Regra*
1	$P \wedge Q$	(premissa)
2	$P \rightarrow R$	(premissa)
3	$(Q \wedge R) \rightarrow S$	(premissa)

Regras de inferência: Exemplo

- Prove S

Passo	*Fórmula*	*Regra*
1	$P \wedge Q$	(premissa)
2	$P \rightarrow R$	(premissa)
3	$(Q \wedge R) \rightarrow S$	(premissa)
4	P	eliminação de E em 1

Lógica Proposicional e Inferência

Regras de inferência: Exemplo

- Prove S

Passo	*Fórmula*	*Regra*
1	$P \wedge Q$	(premissa)
2	$P \rightarrow R$	(premissa)
3	$(Q \wedge R) \rightarrow S$	(premissa)
4	P	eliminação de E em 1
5	R	modus ponens em 4 e 2

Regras de inferência: Exemplo

- Prove S

Passo	*Fórmula*	*Regra*
1	$P \wedge Q$	(premissa)
2	$P \rightarrow R$	(premissa)
3	$(Q \wedge R) \rightarrow S$	(premissa)
4	P	eliminação de E em 1
5	R	modus ponens em 4 e 2
6	Q	eliminação de E em 1

Regras de inferência: Exemplo

- Prove S

Passo	*Fórmula*	*Regra*
1	$P \land Q$	(premissa)
2	$P \rightarrow R$	(premissa)
3	$(Q \land R) \rightarrow S$	(premissa)
4	P	eliminação de E em 1
5	R	modus ponens em 4 e 2
6	Q	eliminação de E em 1
7	$Q \land R$	introdução de E em 5 e 6

Regras de inferência: Exemplo

- Prove S

Passo	*Fórmula*	*Regra*
1	$P \wedge Q$	(premissa)
2	$P \rightarrow R$	(premissa)
3	$(Q \wedge R) \rightarrow S$	(premissa)
4	P	eliminação de E em 1
5	R	modus ponens em 4 e 2
6	Q	eliminação de E em 1
7	$Q \wedge R$	introdução de E em 5 e 6
8	S	modus ponens em 7 e 3

Mais Regras de inferência: Equivalências lógicas

Mais Regras de inferência: Equivalências lógicas

- Idempotentes:
  - $p \vee p \equiv p$
  - $p \wedge p \equiv p$

Mais Regras de inferência: Equivalências lógicas

- Idempotentes:
  - $p \vee p \equiv p$
  - $p \wedge p \equiv p$
- Associativas
  - $(p \vee q) \vee r \equiv p \vee (q \vee r)$
  - $(p \wedge q) \wedge r \equiv p \wedge (q \wedge r)$

Mais Regras de inferência: Equivalências lógicas

- Idempotentes:
  - $p \vee p \equiv p$
  - $p \wedge p \equiv p$
- Associativas
  - $(p \vee q) \vee r \equiv p \vee (q \vee r)$
  - $(p \wedge q) \wedge r \equiv p \wedge (q \wedge r)$
- Comutativas:
  - $p \vee q \equiv q \vee p$
  - $p \wedge q \equiv q \wedge p$

Mais Regras de inferência: Equivalências lógicas

- Idempotentes:
  - $p \vee p \equiv p$
  - $p \wedge p \equiv p$
- Associativas
  - $(p \vee q) \vee r \equiv p \vee (q \vee r)$
  - $(p \wedge q) \wedge r \equiv p \wedge (q \wedge r)$
- Comutativas:
  - $p \vee q \equiv q \vee p$
  - $p \wedge q \equiv q \wedge p$
- Distributivas:
  - $p \vee (q \wedge r) \equiv (p \vee q) \wedge (p \vee r)$
  - $p \wedge (q \vee r) \equiv (p \wedge q) \vee (p \wedge r)$

Lógica Proposicional e Inferência

Mais Regras de inferência: Equivalências lógicas

- Idempotentes:
  - $p \vee p \equiv p$
  - $p \wedge p \equiv p$
- Associativas
  - $(p \vee q) \vee r \equiv p \vee (q \vee r)$
  - $(p \wedge q) \wedge r \equiv p \wedge (q \wedge r)$
- Comutativas:
  - $p \vee q \equiv q \vee p$
  - $p \wedge q \equiv q \wedge p$
- Distributivas:
  - $p \vee (q \wedge r) \equiv (p \vee q) \wedge (p \vee r)$
  - $p \wedge (q \vee r) \equiv (p \wedge q) \vee (p \wedge r)$
- Identidade:
  - $p \wedge F \equiv F$
  - $p \wedge V \equiv p$
  - $p \vee F \equiv p$
  - $p \vee V \equiv V$

Mais Regras de inferência: Equivalências lógicas

- Eliminação:
  - $\neg\neg p \equiv p$
  - $p \rightarrow q \equiv \neg p \vee q$
  - $p \leftrightarrow q \equiv p \rightarrow q \wedge q \rightarrow p$

Mais Regras de inferência: Equivalências lógicas

- Eliminação:
  - $\neg\neg p \equiv p$
  - $p \rightarrow q \equiv \neg p \vee q$
  - $p \leftrightarrow q \equiv p \rightarrow q \wedge q \rightarrow p$
- Complementares:
  - $p \vee \neg p \equiv V$
  - $p \wedge \neg p \equiv F$
  - $\neg F \equiv V$
  - $\neg V \equiv F$

Mais Regras de inferência: Equivalências lógicas

- Eliminação:
  - $\neg\neg p \equiv p$
  - $p \rightarrow q \equiv \neg p \vee q$
  - $p \leftrightarrow q \equiv p \rightarrow q \wedge q \rightarrow p$
- Complementares:
  - $p \vee \neg p \equiv V$
  - $p \wedge \neg p \equiv F$
  - $\neg F \equiv V$
  - $\neg V \equiv F$
- Contraposição:
  - $p \rightarrow q \equiv \neg q \rightarrow \neg p$

Lógica Proposicional e Inferência

Mais Regras de inferência: Equivalências lógicas

- Eliminação:
  - $\neg\neg p \equiv p$
  - $p \rightarrow q \equiv \neg p \vee q$
  - $p \leftrightarrow q \equiv p \rightarrow q \wedge q \rightarrow p$
- Complementares:
  - $p \vee \neg p \equiv V$
  - $p \wedge \neg p \equiv F$
  - $\neg F \equiv V$
  - $\neg V \equiv F$
- Contraposição:
  - $p \rightarrow q \equiv \neg q \rightarrow \neg p$
- De De Morgan:
  - $\neg(p \vee q) \equiv \neg p \wedge \neg q$
  - $\neg(p \wedge q) \equiv \neg p \vee \neg q$

Lógica Proposicional e Inferência

Mais Regras de inferência: Equivalências lógicas

- Eliminação:
  - $\neg\neg p \equiv p$
  - $p \rightarrow q \equiv \neg p \vee q$
  - $p \leftrightarrow q \equiv p \rightarrow q \wedge q \rightarrow p$
- Complementares:
  - $p \vee \neg p \equiv V$
  - $p \wedge \neg p \equiv F$
  - $\neg F \equiv V$
  - $\neg V \equiv F$
- Contraposição:
  - $p \rightarrow q \equiv \neg q \rightarrow \neg p$
- De De Morgan:
  - $\neg(p \vee q) \equiv \neg p \wedge \neg q$
  - $\neg(p \wedge q) \equiv \neg p \vee \neg q$
- Outras:
  - $\neg(p \rightarrow q) \equiv p \wedge \neg q$

Cláusulas Horn

- Bases de conhecimento reais geralmente são restritas quanto à forma de suas sentenças

Cláusulas Horn

- Bases de conhecimento reais geralmente são restritas quanto à forma de suas sentenças
- Algumas contém apenas **Cláusulas Definidas**

Lógica Proposicional e Inferência

Cláusulas Horn

- Bases de conhecimento reais geralmente são restritas quanto à forma de suas sentenças
- Algumas contém apenas **Cláusulas Definidas**
  - Cláusulas contendo a disjunção de literais, dos quais exatamente um é positivo

Lógica Proposicional e Inferência

Cláusulas Horn

- Bases de conhecimento reais geralmente são restritas quanto à forma de suas sentenças
- Algumas contém apenas **Cláusulas Definidas**
  - Cláusulas contendo a disjunção de literais, dos quais exatamente um é positivo
  - Ex: $\neg A \vee \neg B \vee C$

Cláusulas Horn

- Bases de conhecimento reais geralmente são restritas quanto à forma de suas sentenças
- Algumas contém apenas **Cláusulas Definidas**
  - Cláusulas contendo a disjunção de literais, dos quais <u>exatamente</u> um é positivo
  - Ex: $\neg A \vee \neg B \vee C$
- Outras contém apenas **Cláusulas Horn**

Lógica Proposicional e Inferência

Cláusulas Horn

- Bases de conhecimento reais geralmente são restritas quanto à forma de suas sentenças
- Algumas contém apenas **Cláusulas Definidas**
  - Cláusulas contendo a disjunção de literais, dos quais exatamente um é positivo
  - Ex: $\neg A \vee \neg B \vee C$
- Outras contém apenas **Cláusulas Horn**
  - A disjunção de literais dos quais no máximo um é positivo

Lógica Proposicional e Inferência

Cláusulas Horn

- Bases de conhecimento reais geralmente são restritas quanto à forma de suas sentenças
- Algumas contém apenas **Cláusulas Definidas**
  - Cláusulas contendo a disjunção de literais, dos quais exatamente um é positivo
  - Ex: $\neg A \vee \neg B \vee C$
- Outras contém apenas **Cláusulas Horn**
  - A disjunção de literais dos quais no máximo um é positivo
  - Então toda cláusula definida é uma cláusula Horn

Lógica Proposicional e Inferência

Cláusulas?

- Uma cláusula é uma disjunção de literais (positivos ou negativos)

Cláusulas?

- Uma cláusula é uma disjunção de literais (positivos ou negativos)
  - Ex: $A_1 \vee \neg A_2 \vee \ldots \vee A_n$

Lógica Proposicional e Inferência

Cláusulas?

- Uma cláusula é uma disjunção de literais (positivos ou negativos)
  - Ex: $A_1 \vee \neg A_2 \vee \ldots \vee A_n$
- Um único literal também constitui uma cláusula

Cláusulas?

- Uma cláusula é uma disjunção de literais (positivos ou negativos)
  - Ex: $A_1 \vee \neg A_2 \vee \ldots \vee A_n$
- Um único literal também constitui uma cláusula
  - Nesse caso, conhecida como **cláusula unitária**

Cláusulas?

- Uma cláusula é uma disjunção de literais (positivos ou negativos)
  - Ex: $A_1 \vee \neg A_2 \vee \ldots \vee A_n$
- Um único literal também constitui uma cláusula
  - Nesse caso, conhecida como **cláusula unitária**
  - Ex: A_1

Cláusulas Horn

- Uma cláusula Horn constitui então de

Lógica Proposicional e Inferência

Cláusulas Horn

- Uma cláusula Horn constitui então de
  - Símbolo literal positivo

Cláusulas Horn

- Uma cláusula Horn constitui então de
  - Símbolo literal positivo
  - Ou disjunção de literais negativos

Lógica Proposicional e Inferência

Cláusulas Horn

- Uma cláusula Horn constitui então de
  - Símbolo literal positivo
  - Ou disjunção de literais negativos
    - $\neg A_1 \vee \neg A_2 \vee \ldots \vee \neg A_n$

Cláusulas Horn

- Uma cláusula Horn constitui então de
  - Símbolo literal positivo
  - Ou disjunção de literais negativos
    - $\neg A_1 \vee \neg A_2 \vee \ldots \vee \neg A_n$
    - Ou (conjunção de literais positivos) $\Rightarrow$ *falso*

Cláusulas Horn

- Uma cláusula Horn constitui então de
  - Símbolo literal positivo
  - Ou disjunção de literais negativos
    - $\neg A_1 \vee \neg A_2 \vee \ldots \vee \neg A_n$
    - Ou (conjunção de literais positivos) $\Rightarrow$ *falso*
    - $A_1 \wedge A_2 \wedge \ldots \wedge A_n \rightarrow \mathit{falso}$

Cláusulas Horn

- Uma cláusula Horn constitui então de
  - Símbolo literal positivo
  - Ou disjunção de literais negativos
    - $\neg A_1 \vee \neg A_2 \vee \ldots \vee \neg A_n$
    - Ou (conjunção de literais positivos) $\Rightarrow$ *falso*
    - $A_1 \wedge A_2 \wedge \ldots \wedge A_n \rightarrow falso$
  - Ou disjunção de literais negativos com um positivo

Cláusulas Horn

- Uma cláusula Horn constitui então de
  - Símbolo literal positivo
  - Ou disjunção de literais negativos
    - $\neg A_1 \vee \neg A_2 \vee \ldots \vee \neg A_n$
    - Ou (conjunção de literais positivos) $\Rightarrow$ *falso*
    - $A_1 \wedge A_2 \wedge \ldots \wedge A_n \rightarrow \mathit{falso}$
  - Ou disjunção de literais negativos com um positivo
    - $\neg A_1 \vee \neg A_2 \vee \ldots \vee \neg A_n \vee B$

Cláusulas Horn

- Uma cláusula Horn constitui então de
  - Símbolo literal positivo
  - Ou disjunção de literais negativos
    - $\neg A_1 \vee \neg A_2 \vee \ldots \vee \neg A_n$
    - Ou (conjunção de literais positivos) $\Rightarrow$ *falso*
    - $A_1 \wedge A_2 \wedge \ldots \wedge A_n \rightarrow \mathit{falso}$
  - Ou disjunção de literais negativos com um positivo
    - $\neg A_1 \vee \neg A_2 \vee \ldots \vee \neg A_n \vee B$
    - Ou (conjunção de literais positivos) $\Rightarrow$ literal positivo

Lógica Proposicional e Inferência

Cláusulas Horn

- Uma cláusula Horn constitui então de
  - Símbolo literal positivo
  - Ou disjunção de literais negativos
    - $\neg A_1 \vee \neg A_2 \vee \ldots \vee \neg A_n$
    - Ou (conjunção de literais positivos) $\Rightarrow$ *falso*
    - $A_1 \wedge A_2 \wedge \ldots \wedge A_n \rightarrow \mathit{falso}$
  - Ou disjunção de literais negativos com um positivo
    - $\neg A_1 \vee \neg A_2 \vee \ldots \vee \neg A_n \vee B$
    - Ou (conjunção de literais positivos) $\Rightarrow$ literal positivo
    - $A_1 \wedge A_2 \wedge \ldots \wedge A_n \rightarrow B$

Lógica Proposicional e Inferência

Cláusulas Horn – Modus Ponens

- Modus Ponens para a forma Horn:

$$\frac{\alpha_1 \wedge \ldots \wedge \alpha_n \rightarrow \beta \qquad \alpha_1, \ldots, \alpha_n}{\beta} \quad \text{ou} \quad \frac{\alpha_1 \wedge \ldots \wedge \alpha_n \rightarrow \mathit{falso} \qquad \alpha_1, \ldots, \alpha_n}{\mathit{falso}}$$

Lógica Proposicional e Inferência

Cláusulas Horn – Modus Ponens

- Modus Ponens para a forma Horn:

$$\frac{\alpha_1 \wedge \ldots \wedge \alpha_n \rightarrow \beta \qquad \alpha_1, \ldots, \alpha_n}{\beta} \quad \text{ou} \quad \frac{\alpha_1 \wedge \ldots \wedge \alpha_n \rightarrow \mathit{falso} \qquad \alpha_1, \ldots, \alpha_n}{\mathit{falso}}$$

- Para uma base de Cláusulas Definidas, modus ponens é derivado por *Forward Chaining* ou *Backward Chaining* em tempo linear com o tamanho da base.

Cláusulas Horn: *Forward Chaining*

- *Forward Chaining* determina se uma única proposição (Q – a *query*) é acarretada por uma base de cláusulas definidas.

Cláusulas Horn: *Forward Chaining*

- *Forward Chaining* determina se uma única proposição (Q – a *query*) é acarretada por uma base de cláusulas definidas.
- Funcionamento:

Cláusulas Horn: *Forward Chaining*

- *Forward Chaining* determina se uma única proposição (Q – a *query*) é acarretada por uma base de cláusulas definidas.
- Funcionamento:
    - Comece com fatos conhecidos (literais positivos)

Lógica Proposicional e Inferência

Cláusulas Horn: *Forward Chaining*

- *Forward Chaining* determina se uma única proposição (Q – a *query*) é acarretada por uma base de cláusulas definidas.
- Funcionamento:
  - Comece com fatos conhecidos (literais positivos)
  - Se todas as premissas de uma implicação são conhecidas, adicione sua conclusão ao conjunto de fatos conhecidos

Cláusulas Horn: *Forward Chaining*

- *Forward Chaining* determina se uma única proposição (Q – a *query*) é acarretada por uma base de cláusulas definidas.
- Funcionamento:
  - Comece com fatos conhecidos (literais positivos)
  - Se todas as premissas de uma implicação são conhecidas, adicione sua conclusão ao conjunto de fatos conhecidos
    - Se A e B são conhecidas, e $(A \wedge B) \rightarrow C$ está na base, então C é adicionado

Lógica Proposicional e Inferência

Cláusulas Horn: *Forward Chaining*

- *Forward Chaining* determina se uma única proposição (Q – a *query*) é acarretada por uma base de cláusulas definidas.
- Funcionamento:
  - Comece com fatos conhecidos (literais positivos)
  - Se todas as premissas de uma implicação são conhecidas, adicione sua conclusão ao conjunto de fatos conhecidos
    - Se A e B são conhecidas, e $(A \wedge B) \rightarrow C$ está na base, então C é adicionado
  - O processo continua até Q ser adicionada ou até não podermos mais inferir nada

Forward Chaining: Algoritmo

Função *FORWARD_CHAINING(BC, q)*: **boolean**

    cont ← número de símbolos em cada premissa
    inferida ← inicializada com falso para todo símbolo
    agenda ← símbolos verdadeiros na BC

    enquanto agenda não estiver vazia faça
        p ← retira o primeiro elemento de agenda
        se p = q então retorna verdadeiro
        se inferida[p] = falso então
            inferida[p] ← verdadeiro
            para cada cláusula c em que p ∈ PREMISSAS(c) faça
                decremente cont[c]
                se cont[c] = 0 então
                    Adicione a conclusão de c à agenda

    retorna falso;

Lógica Proposicional e Inferência

Forward Chaining: Algoritmo

Função *FORWARD_CHAINING(BC, q)*: **boolean**

```
cont ← número de símbolos em cada premissa
inferida ← inicializada com falso para todo símbolo
agenda ← símbolos verdadeiros na BC

enquanto agenda não estiver vazia faça
    p ← retira o primeiro elemento de agenda
    se p = q então  retorna verdadeiro
    se inferida[p] = falso então
        inferida[p] ← verdadeiro
        para cada cláusula c em que p ∈ PREMISSAS(c) faça
            decremente cont[c]
            se cont[c] = 0 então
                Adicione a conclusão de c à agenda

retorna falso;
```

query (símbolo proposicional) a ser provada

Forward Chaining: Algoritmo

Função *FORWARD_CHAINING(BC, q)*: **boolean**

cont $\leftarrow$ número de símbolos em cada premissa
inferida $\leftarrow$ inicializada com *falso* para todo símbolo
agenda $\leftarrow$ símbolos verdadeiros na *BC*

tabela onde *cont[c]* é o número de símbolos na premissa de *c*

enquanto *agenda não estiver vazia* **faça**
  p $\leftarrow$ retira o primeiro elemento de *agenda*
  se $p = q$ **então** **retorna** *verdadeiro*
  se $inferida[p] = falso$ **então**
    inferida[p] $\leftarrow$ *verdadeiro*
    para cada *cláusula c em que* $p \in PREMISSAS(c)$ **faça**
      decremente $cont[c]$
      se $cont[c] = 0$ **então**
        Adicione a conclusão de c à *agenda*

retorna *falso*;

Forward Chaining: Algoritmo

Função *FORWARD_CHAINING(BC, q)*: **boolean**

- *cont* $\leftarrow$ número de símbolos em cada premissa
- *inferida* $\leftarrow$ inicializada com *falso* para todo símbolo
- *agenda* $\leftarrow$ símbolos verdadeiros na *BC*
- **enquanto** *agenda não estiver vazia* **faça**
  - *p* $\leftarrow$ retira o primeiro elemento de *agenda*
  - **se** $p = q$ **então** **retorna** *verdadeiro*
  - **se** $inferida[p] = falso$ **então**
    - *inferida[p]* $\leftarrow$ *verdadeiro*
    - **para cada** *cláusula c em que* $p \in PREMISSAS(c)$ **faça**
      - decremente $cont[c]$
      - **se** $cont[c] = 0$ **então**
        - Adicione a conclusão de *c* à *agenda*
- **retorna** *falso*;

tabela onde *inferida*$[s]$ é inicialmente *falso* para todos os símbolos

Forward Chaining: Algoritmo

Função *FORWARD_CHAINING(BC, q)*: **boolean**

cont ← número de símbolos em cada premissa
inferida ← inicializada com *falso* para todo símbolo
agenda ← símbolos verdadeiros na *BC*

enquanto *agenda não estiver vazia* **faça**
  p ← retira o primeiro elemento de *agenda*
  se $p = q$ **então** **retorna** *verdadeiro*
  se *inferida*$[p]$ = *falso* **então**
    inferida[p] ← *verdadeiro*
    para cada *cláusula c em que* $p \in PREMISSAS(c)$ **faça**
      decremente *cont*$[c]$
      se *cont*$[c] = 0$ **então**
        Adicione a conclusão de *c* à *agenda*

retorna *falso*;

fila de símbolos. Inicialmente símbolos conhecidamente verdadeiros em *BC*

Lógica Proposicional e Inferência

Forward Chaining: Algoritmo

Função *FORWARD_CHAINING(BC, q)*: **boolean**

    cont ← número de símbolos em cada premissa
    inferida ← inicializada com *falso* para todo símbolo
    agenda ← símbolos verdadeiros na *BC*

    enquanto *agenda não estiver vazia* **faça**
        p ← retira o primeiro elemento de *agenda*
        se $p = q$ **então** **retorna** *verdadeiro*
        se *inferida*$[p]$ = *falso* **então**
            inferida[p] ← *verdadeiro*
            para cada *cláusula c em que* $p \in$ *PREMISSAS*(c) **faça**
                decremente *cont*$[c]$
                se *cont*$[c] = 0$ **então**
                    Adicione a conclusão de *c* à *agenda*

    retorna *falso*;

verificamos toda cláusula $c \in BC$ em cujas premissas p aparecer

Forward Chaining: Exemplo

- BC:

SE causa danos físicos, deve ser evitado	$F \rightarrow E$
SE é perigoso E morde, causa danos físicos	$P \wedge M \rightarrow F$
SE tem dentes E é perigoso, morde	$D \wedge P \rightarrow M$
SE é agressivo E causa danos físicos, é perigoso	$A \wedge F \rightarrow P$
SE é agressivo E tem dentes, é perigoso	$A \wedge D \rightarrow P$
É agressivo	A
Tem dentes	D

Lógica Proposicional e Inferência

Forward Chaining: Exemplo

- BC:

SE causa danos físicos, deve ser evitado	$F \rightarrow E$
SE é perigoso E morde, causa danos físicos	$P \wedge M \rightarrow F$
SE tem dentes E é perigoso, morde	$D \wedge P \rightarrow M$
SE é agressivo E causa danos físicos, é perigoso	$A \wedge F \rightarrow P$
SE é agressivo E tem dentes, é perigoso	$A \wedge D \rightarrow P$
É agressivo	A
Tem dentes	D

- Query: Deve ser evitado ($E = verdadeiro$)?

Forward Chaining: Algoritmo

Função *FORWARD_CHAINING(BC, q)*: **boolean**
- *cont* $\leftarrow$ número de símbolos em cada premissa
- *inferida* $\leftarrow$ inicializada com *falso* para todo símbolo
- *agenda* $\leftarrow$ símbolos verdadeiros na *BC*
- **enquanto** *agenda não estiver vazia* **faça**
  - $p \leftarrow$ retira o primeiro elemento de *agenda*
  - **se** $p = q$ **então**
    - **retorna** *verdadeiro*
  - **se** $inferida[p] = falso$ **então**
    - *inferida[p]* $\leftarrow$ *verdadeiro*
    - **para cada** *cláusula c em que* $p \in PREMISSAS(c)$ **faça**
      - decremente $cont[c]$
      - **se** $cont[c] = 0$ **então**
        - Adicione a conclusão de *c* à *agenda*
- **retorna** *falso*;

BC:
1: $F \rightarrow E$
2: $P \wedge M \rightarrow F$
3: $D \wedge P \rightarrow M$
4: $A \wedge F \rightarrow P$
5: $A \wedge D \rightarrow P$
6: A
7: D

q: E

Lógica Proposicional e Inferência

Forward Chaining: Algoritmo

Função *FORWARD_CHAINING(BC, q)*: **boolean**

  cont ← número de símbolos em cada premissa
  inferida ← inicializada com *falso* para todo símbolo
  agenda ← símbolos verdadeiros na *BC*

  enquanto *agenda não estiver vazia* **faça**
    p ← retira o primeiro elemento de *agenda*
    se $p = q$ **então**
      retorna *verdadeiro*
    se $inferida[p] = falso$ **então**
      inferida[p] ← *verdadeiro*
      para cada *cláusula c em que* $p \in PREMISSAS(c)$ **faça**
        decremente $cont[c]$
        se $cont[c] = 0$ **então**
          Adicione a conclusão de *c* à *agenda*

  retorna *falso*;

BC:
1: $F \rightarrow E$
2: $P \wedge M \rightarrow F$
3: $D \wedge P \rightarrow M$
4: $A \wedge F \rightarrow P$
5: $A \wedge D \rightarrow P$
6: A
7: D

q: E

cont:

1	2	3	4	5	6	7
1	2	2	2	2	0	0

Forward Chaining: Algoritmo

Função *FORWARD_CHAINING(BC, q)*: **boolean**

cont ← número de símbolos em cada premissa
inferida ← inicializada com *falso* para todo símbolo
agenda ← símbolos verdadeiros na *BC*

enquanto *agenda não estiver vazia* **faça**
 p ← retira o primeiro elemento de *agenda*
 se $p = q$ **então**
  retorna *verdadeiro*
 se *inferida*[*p*] = *falso* **então**
  inferida[p] ← *verdadeiro*
  para cada *cláusula c em que* $p \in PREMISSAS(c)$ **faça**
   decremente *cont*[*c*]
   se *cont*[*c*] = 0 **então**
    Adicione a conclusão de *c* à *agenda*

retorna *falso*;

BC:

1:	$F \rightarrow E$
2:	$P \wedge M \rightarrow F$
3:	$D \wedge P \rightarrow M$
4:	$A \wedge F \rightarrow P$
5:	$A \wedge D \rightarrow P$
6:	A
7:	D

q: E

cont:

1	2	3	4	5	6	7
1	2	2	2	2	0	0

inferida:

A	D	E	F	M	P
F	F	F	F	F	F

Lógica Proposicional e Inferência

Forward Chaining: Algoritmo

Função *FORWARD_CHAINING(BC, q)*: **boolean**

    cont ← número de símbolos em cada premissa
    inferida ← inicializada com falso para todo símbolo
    agenda ← símbolos verdadeiros na BC

    enquanto agenda não estiver vazia faça
        p ← retira o primeiro elemento de agenda
        se p = q então
            retorna verdadeiro
        se inferida[p] = falso então
            inferida[p] ← verdadeiro
            para cada cláusula c em que p ∈ PREMISSAS(c) faça
                decremente cont[c]
                se cont[c] = 0 então
                    Adicione a conclusão de c à agenda
    retorna falso;

BC:
1: $F \rightarrow E$
2: $P \wedge M \rightarrow F$
3: $D \wedge P \rightarrow M$
4: $A \wedge F \rightarrow P$
5: $A \wedge D \rightarrow P$
6: A
7: D

q: E

cont:

1	2	3	4	5	6	7
1	2	2	2	2	0	0

inferida:

A	*D*	*E*	*F*	*M*	*P*
F	*F*	*F*	*F*	*F*	*F*

agenda:

A	*D*

Forward Chaining: Algoritmo

Função *FORWARD_CHAINING(BC, q)*: **boolean**

- *cont* ← número de símbolos em cada premissa
- *inferida* ← inicializada com *falso* para todo símbolo
- *agenda* ← símbolos verdadeiros na *BC*
- **enquanto** *agenda não estiver vazia* **faça**
  - *p* ← retira o primeiro elemento de *agenda*
  - **se** $p = q$ **então**
    - **retorna** *verdadeiro*
  - **se** $inferida[p] = falso$ **então**
    - *inferida[p]* ← *verdadeiro*
    - **para cada** *cláusula c em que* $p \in PREMISSAS(c)$ **faça**
      - decremente $cont[c]$
      - **se** $cont[c] = 0$ **então**
        - Adicione a conclusão de *c* à *agenda*
- **retorna** *falso*;

BC:

1: $F \rightarrow E$
2: $P \wedge M \rightarrow F$
3: $D \wedge P \rightarrow M$
4: $A \wedge F \rightarrow P$
5: $A \wedge D \rightarrow P$
6: A
7: D

q: E

cont:

1	2	3	4	5	6	7
1	2	2	2	2	0	0

inferida:

A	D	E	F	M	P
F	F	F	F	F	F

agenda:

A	D

Forward Chaining: Algoritmo

Função *FORWARD_CHAINING(BC, q)*: **boolean**

- *cont* ← número de símbolos em cada premissa
- *inferida* ← inicializada com *falso* para todo símbolo
- *agenda* ← símbolos verdadeiros na *BC*
- **enquanto** *agenda não estiver vazia* **faça**
  - *p* ← retira o primeiro elemento de *agenda*
  - **se** $p = q$ **então**
    - **retorna** *verdadeiro*
  - **se** *inferida*$[p]$ $=$ *falso* **então**
    - *inferida[p]* ← *verdadeiro*
    - **para cada** *cláusula c em que* $p \in PREMISSAS(c)$ **faça**
      - decremente *cont*$[c]$
      - **se** *cont*$[c] = 0$ **então**
        - Adicione a conclusão de *c* à *agenda*
- **retorna** *falso*;

BC:

1:	$F \rightarrow E$
2:	$P \wedge M \rightarrow F$
3:	$D \wedge P \rightarrow M$
4:	$A \wedge F \rightarrow P$
5:	$A \wedge D \rightarrow P$
6:	A
7:	D

q: E p: A

cont:

1	2	3	4	5	6	7
1	2	2	2	2	0	0

inferida:

A	D	E	F	M	P
F	F	F	F	F	F

agenda: D

Lógica Proposicional e Inferência

Forward Chaining: Algoritmo

Função *FORWARD_CHAINING(BC, q)*: **boolean**

```
cont ← número de símbolos em cada premissa
inferida ← inicializada com falso para todo símbolo
agenda ← símbolos verdadeiros na BC

enquanto agenda não estiver vazia faça
    p ← retira o primeiro elemento de agenda
    se p = q então
        retorna verdadeiro
    se inferida[p] = falso então
        inferida[p] ← verdadeiro
        para cada cláusula c em que
        p ∈ PREMISSAS(c) faça
            decremente cont[c]
            se cont[c] = 0 então
                Adicione a conclusão de c à agenda
retorna falso;
```

BC:

1:	$F \rightarrow E$
2:	$P \wedge M \rightarrow F$
3:	$D \wedge P \rightarrow M$
4:	$A \wedge F \rightarrow P$
5:	$A \wedge D \rightarrow P$
6:	A
7:	D

q: E p: A

cont:

1	2	3	4	5	6	7
1	2	2	2	2	0	0

inferida:

A	*D*	*E*	*F*	*M*	*P*
F	F	F	F	F	F

agenda: $\boxed{D}$

Forward Chaining: Algoritmo

Função *FORWARD_CHAINING(BC, q)*: **boolean**
  cont ← número de símbolos em cada premissa
  inferida ← inicializada com *falso* para todo símbolo
  agenda ← símbolos verdadeiros na *BC*

  enquanto *agenda não estiver vazia* **faça**
    p ← retira o primeiro elemento de *agenda*
    se $p = q$ **então**
      retorna *verdadeiro*
    se $inferida[p] = falso$ **então**
      $inferida[p] \leftarrow verdadeiro$
      para cada *cláusula c em que* $p \in PREMISSAS(c)$ **faça**
        decremente $cont[c]$
        se $cont[c] = 0$ **então**
          Adicione a conclusão de *c* à *agenda*
  retorna *falso*;

BC:
1: $F \rightarrow E$
2: $P \wedge M \rightarrow F$
3: $D \wedge P \rightarrow M$
4: $A \wedge F \rightarrow P$
5: $A \wedge D \rightarrow P$
6: A
7: D

q: E p: A

cont:

1	2	3	4	5	6	7
1	2	2	2	2	0	0

inferida:

A	D	E	F	M	P
F	F	F	F	F	F

agenda: D

Forward Chaining: Algoritmo

Função *FORWARD_CHAINING(BC, q)*: **boolean**

```
cont ← número de símbolos em cada premissa
inferida ← inicializada com falso para todo símbolo
agenda ← símbolos verdadeiros na BC

enquanto agenda não estiver vazia faça
    p ← retira o primeiro elemento de agenda
    se p = q então
        retorna verdadeiro
    se inferida[p] = falso então
        inferida[p] ← verdadeiro
        para cada cláusula c em que
        p ∈ PREMISSAS(c) faça
            decremente cont[c]
            se cont[c] = 0 então
                Adicione a conclusão de c à agenda
retorna falso;
```

BC:

1: $F \rightarrow E$
2: $P \wedge M \rightarrow F$
3: $D \wedge P \rightarrow M$
4: $A \wedge F \rightarrow P$
5: $A \wedge D \rightarrow P$
6: A
7: D

q: E p: A

cont:

	1	2	3	4	5	6	7
cont	1	2	2	2	2	0	0

inferida:

	A	D	E	F	M	P
inferida	V	F	F	F	F	F

agenda: D

Lógica Proposicional e Inferência

Forward Chaining: Algoritmo

Função *FORWARD_CHAINING(BC, q)*: **boolean**
  cont ← número de símbolos em cada premissa
  inferida ← inicializada com *falso* para todo símbolo
  agenda ← símbolos verdadeiros na *BC*

  enquanto *agenda não estiver vazia* **faça**
    p ← retira o primeiro elemento de *agenda*
    se $p = q$ **então**
      retorna *verdadeiro*
    se *inferida*[*p*] = *falso* **então**
      inferida[p] ← *verdadeiro*
      para cada cláusula *c* em que
      $p \in PREMISSAS(c)$ **faça**
        decremente *cont*[*c*]
        se *cont*[*c*] = 0 **então**
          Adicione a conclusão de *c* à *agenda*
  retorna *falso*;

BC:
1: $F \rightarrow E$
2: $P \wedge M \rightarrow F$
3: $D \wedge P \rightarrow M$
4: $A \wedge F \rightarrow P$
5: $A \wedge D \rightarrow P$
6: A
7: D

q: E p: A c: 4

cont:

1	2	3	4	5	6	7
1	2	2	2	2	0	0

inferida:

A	*D*	*E*	*F*	*M*	*P*
V	*F*	*F*	*F*	*F*	*F*

agenda: | *D* |

Forward Chaining: Algoritmo

Função *FORWARD_CHAINING(BC, q)*: **boolean**

- *cont* ← número de símbolos em cada premissa
- *inferida* ← inicializada com *falso* para todo símbolo
- *agenda* ← símbolos verdadeiros na *BC*
- **enquanto** *agenda não estiver vazia* **faça**
  - *p* ← retira o primeiro elemento de *agenda*
  - **se** $p = q$ **então**
    - **retorna** *verdadeiro*
  - **se** *inferida*[*p*] = *falso* **então**
    - *inferida[p]* ← *verdadeiro*
    - **para cada** *cláusula c em que* $p \in PREMISSAS(c)$ **faça**
      - decremente *cont*[*c*]
      - **se** *cont*[*c*] = 0 **então**
        - Adicione a conclusão de *c* à *agenda*
- **retorna** *falso*;

BC:

1: $F \rightarrow E$
2: $P \wedge M \rightarrow F$
3: $D \wedge P \rightarrow M$
4: $A \wedge F \rightarrow P$
5: $A \wedge D \rightarrow P$
6: A
7: D

q: E p: A c: 4

cont:

1	2	3	4	5	6	7
1	2	2	1	2	0	0

inferida:

A	*D*	*E*	*F*	*M*	*P*
V	*F*	*F*	*F*	*F*	*F*

agenda: *D*

Forward Chaining: Algoritmo

Função *FORWARD_CHAINING(BC, q)*: **boolean**
  cont ← número de símbolos em cada premissa
  inferida ← inicializada com *falso* para todo símbolo
  agenda ← símbolos verdadeiros na *BC*

  enquanto *agenda não estiver vazia* **faça**
    p ← retira o primeiro elemento de *agenda*
    se $p = q$ **então**
      retorna *verdadeiro*
    se $inferida[p] = falso$ **então**
      inferida[p] ← *verdadeiro*
      para cada *cláusula c em que*
      $p \in PREMISSAS(c)$ **faça**
        decremente $cont[c]$
        se $cont[c] = 0$ **então**
          Adicione a conclusão de *c* à *agenda*
  retorna *falso*;

BC:
1: $F \rightarrow E$
2: $P \wedge M \rightarrow F$
3: $D \wedge P \rightarrow M$
4: $A \wedge F \rightarrow P$
5: $A \wedge D \rightarrow P$
6: A
7: D

q: E p: A c: 4

cont:

1	2	3	4	5	6	7
1	2	2	1	2	0	0

inferida:

A	D	E	F	M	P
V	F	F	F	F	F

agenda: D

Forward Chaining: Algoritmo

Função *FORWARD_CHAINING(BC, q)*: **boolean**
- *cont* $\leftarrow$ número de símbolos em cada premissa
- *inferida* $\leftarrow$ inicializada com *falso* para todo símbolo
- *agenda* $\leftarrow$ símbolos verdadeiros na *BC*
- **enquanto** *agenda não estiver vazia* **faça**
  - $p \leftarrow$ retira o primeiro elemento de *agenda*
  - **se** $p = q$ **então**
    - **retorna** *verdadeiro*
  - **se** $inferida[p] = falso$ **então**
    - *inferida[p]* $\leftarrow$ *verdadeiro*
    - **para cada** *cláusula c em que* $p \in PREMISSAS(c)$ **faça**
      - decremente $cont[c]$
      - **se** $cont[c] = 0$ **então**
        - Adicione a conclusão de *c* à *agenda*
- **retorna** *falso*;

BC:
1: $F \rightarrow E$
2: $P \wedge M \rightarrow F$
3: $D \wedge P \rightarrow M$
4: $A \wedge F \rightarrow P$
5: $A \wedge D \rightarrow P$
6: A
7: D

q: E p: A c: 5

cont:

1	2	3	4	5	6	7
1	2	2	1	2	0	0

inferida:

A	D	E	F	M	P
V	F	F	F	F	F

agenda: D

Forward Chaining: Algoritmo

Função *FORWARD_CHAINING(BC, q)*: **boolean**

cont ← número de símbolos em cada premissa
inferida ← inicializada com *falso* para todo símbolo
agenda ← símbolos verdadeiros na *BC*

enquanto *agenda não estiver vazia* **faça**
- *p* ← retira o primeiro elemento de *agenda*
- **se** $p = q$ **então**
  - **retorna** *verdadeiro*
- **se** $inferida[p] = falso$ **então**
  - *inferida[p]* ← *verdadeiro*
  - **para cada** *cláusula c em que* $p \in PREMISSAS(c)$ **faça**
    - decremente $cont[c]$
    - **se** $cont[c] = 0$ **então**
      - Adicione a conclusão de *c* à *agenda*

retorna *falso*;

BC:
1: $F \rightarrow E$
2: $P \wedge M \rightarrow F$
3: $D \wedge P \rightarrow M$
4: $A \wedge F \rightarrow P$
5: $A \wedge D \rightarrow P$
6: A
7: D

q: E p: A c: 5

cont:

1	2	3	4	5	6	7
1	2	2	1	1	0	0

inferida:

A	D	E	F	M	P
V	F	F	F	F	F

agenda: D

Lógica Proposicional e Inferência

Forward Chaining: Algoritmo

Função *FORWARD_CHAINING(BC, q)*: **boolean**
- *cont* ← número de símbolos em cada premissa
- *inferida* ← inicializada com *falso* para todo símbolo
- *agenda* ← símbolos verdadeiros na *BC*
- **enquanto** *agenda não estiver vazia* **faça**
  - *p* ← retira o primeiro elemento de *agenda*
  - **se** $p = q$ **então**
    - **retorna** *verdadeiro*
  - **se** $inferida[p] = falso$ **então**
    - *inferida[p]* ← *verdadeiro*
    - **para cada** *cláusula c em que* $p \in PREMISSAS(c)$ **faça**
      - decremente $cont[c]$
      - **se** $cont[c] = 0$ **então**
        - Adicione a conclusão de *c* à *agenda*
- **retorna** *falso*;

BC:

1:	$F \rightarrow E$
2:	$P \wedge M \rightarrow F$
3:	$D \wedge P \rightarrow M$
4:	$A \wedge F \rightarrow P$
5:	$A \wedge D \rightarrow P$
6:	A
7:	D

q: E p: A c: 5

cont:

1	2	3	4	5	6	7
1	2	2	1	1	0	0

inferida:

A	D	E	F	M	P
V	F	F	F	F	F

agenda: D

Forward Chaining: Algoritmo

Função *FORWARD_CHAINING(BC, q)*: **boolean**
  cont ← número de símbolos em cada premissa
  inferida ← inicializada com *falso* para todo símbolo
  agenda ← símbolos verdadeiros na *BC*

  enquanto *agenda não estiver vazia* **faça**
    p ← retira o primeiro elemento de *agenda*
    se $p = q$ **então**
      retorna *verdadeiro*
    se *inferida*[*p*] = *falso* **então**
      inferida[p] ← *verdadeiro*
      para cada *cláusula c em que*
      $p \in PREMISSAS(c)$ **faça**
        decremente *cont*[*c*]
        se *cont*[*c*] = 0 **então**
          Adicione a conclusão de *c* à *agenda*
  retorna *falso*;

BC:
1: $F \rightarrow E$
2: $P \wedge M \rightarrow F$
3: $D \wedge P \rightarrow M$
4: $A \wedge F \rightarrow P$
5: $A \wedge D \rightarrow P$
6: A
7: D

q: E p: A c: 5

cont:

1	2	3	4	5	6	7
1	2	2	1	1	0	0

inferida:

A	D	E	F	M	P
V	F	F	F	F	F

agenda: D

Lógica Proposicional e Inferência

Forward Chaining: Algoritmo

Função *FORWARD_CHAINING(BC, q)*: **boolean**

- *cont* ← número de símbolos em cada premissa
- *inferida* ← inicializada com *falso* para todo símbolo
- *agenda* ← símbolos verdadeiros na *BC*
- **enquanto** *agenda não estiver vazia* **faça**
  - *p* ← retira o primeiro elemento de *agenda*
  - **se** $p = q$ **então**
    - **retorna** *verdadeiro*
  - **se** *inferida*$[p]$ = *falso* **então**
    - *inferida[p]* ← *verdadeiro*
    - **para cada** *cláusula c em que* $p \in PREMISSAS(c)$ **faça**
      - decremente *cont*$[c]$
      - **se** *cont*$[c] = 0$ **então**
        - Adicione a conclusão de *c* à *agenda*
- **retorna** *falso*;

BC:
1: $F \rightarrow E$
2: $P \wedge M \rightarrow F$
3: $D \wedge P \rightarrow M$
4: $A \wedge F \rightarrow P$
5: $A \wedge D \rightarrow P$
6: A
7: D

q: E p: A c: 5

cont:

1	2	3	4	5	6	7
1	2	2	1	1	0	0

inferida:

A	D	E	F	M	P
V	F	F	F	F	F

agenda: $\boxed{D}$

Lógica Proposicional e Inferência

Forward Chaining: Algoritmo

Função *FORWARD_CHAINING(BC, q)*: **boolean**

cont ← número de símbolos em cada premissa
inferida ← inicializada com *falso* para todo símbolo
agenda ← símbolos verdadeiros na *BC*

enquanto *agenda não estiver vazia* **faça**
  p ← retira o primeiro elemento de *agenda*
  se $p = q$ **então**
    retorna *verdadeiro*
  se $inferida[p] = falso$ **então**
    inferida[p] ← *verdadeiro*
    para cada *cláusula c em que* $p \in PREMISSAS(c)$ **faça**
      decremente $cont[c]$
      se $cont[c] = 0$ **então**
        Adicione a conclusão de *c* à *agenda*

retorna *falso*;

BC:
1: $F \rightarrow E$
2: $P \wedge M \rightarrow F$
3: $D \wedge P \rightarrow M$
4: $A \wedge F \rightarrow P$
5: $A \wedge D \rightarrow P$
6: A
7: D

q: E p: D c: 5

cont:

1	2	3	4	5	6	7
1	2	2	1	1	0	0

inferida:

A	D	E	F	M	P
V	F	F	F	F	F

agenda: (vazia)

Forward Chaining: Algoritmo

Função *FORWARD_CHAINING(BC, q)*: **boolean**
- *cont* ← número de símbolos em cada premissa
- *inferida* ← inicializada com *falso* para todo símbolo
- *agenda* ← símbolos verdadeiros na *BC*
- **enquanto** *agenda não estiver vazia* **faça**
  - *p* ← retira o primeiro elemento de *agenda*
  - **se** $p = q$ **então**
    - **retorna** *verdadeiro*
  - **se** *inferida*[*p*] = *falso* **então**
    - *inferida[p]* ← *verdadeiro*
    - **para cada** *cláusula c em que* $p \in PREMISSAS(c)$ **faça**
      - decremente *cont*[*c*]
      - **se** *cont*[*c*] = 0 **então**
        - Adicione a conclusão de *c* à *agenda*
- **retorna** *falso*;

BC:
1: $F \rightarrow E$
2: $P \wedge M \rightarrow F$
3: $D \wedge P \rightarrow M$
4: $A \wedge F \rightarrow P$
5: $A \wedge D \rightarrow P$
6: A
7: D

q: E p: D c: 5

cont:

1	2	3	4	5	6	7
1	2	2	1	1	0	0

inferida:

A	D	E	F	M	P
V	F	F	F	F	F

agenda: (vazia)

Forward Chaining: Algoritmo

Função *FORWARD_CHAINING(BC, q)*: **boolean**
  cont ← número de símbolos em cada premissa
  inferida ← inicializada com *falso* para todo símbolo
  agenda ← símbolos verdadeiros na *BC*

  enquanto *agenda não estiver vazia* **faça**
    $p \leftarrow$ retira o primeiro elemento de *agenda*
    se $p = q$ **então**
      retorna *verdadeiro*
    se $inferida[p] = falso$ **então**
      $inferida[p] \leftarrow verdadeiro$
      para cada *cláusula c em que* $p \in PREMISSAS(c)$ **faça**
        decremente $cont[c]$
        se $cont[c] = 0$ **então**
          Adicione a conclusão de *c* à *agenda*
  retorna *falso*;

BC:
1: $F \rightarrow E$
2: $P \wedge M \rightarrow F$
3: $D \wedge P \rightarrow M$
4: $A \wedge F \rightarrow P$
5: $A \wedge D \rightarrow P$
6: A
7: D

q: E p: D c: 5

cont:

1	2	3	4	5	6	7
1	2	2	1	1	0	0

inferida:

A	D	E	F	M	P
V	F	F	F	F	F

agenda: (vazia)

Forward Chaining: Algoritmo

Função *FORWARD_CHAINING(BC, q)*: **boolean**
 cont ← número de símbolos em cada premissa
 inferida ← inicializada com *falso* para todo símbolo
 agenda ← símbolos verdadeiros na *BC*

 enquanto *agenda não estiver vazia* **faça**
  p ← retira o primeiro elemento de *agenda*
  se $p = q$ **então**
   retorna *verdadeiro*
  se $inferida[p] = falso$ **então**
   inferida[p] ← *verdadeiro*
   para cada *cláusula c em que* $p \in PREMISSAS(c)$ **faça**
    decremente $cont[c]$
    se $cont[c] = 0$ **então**
     Adicione a conclusão de c à *agenda*
 retorna *falso*;

BC:
1: $F \rightarrow E$
2: $P \wedge M \rightarrow F$
3: $D \wedge P \rightarrow M$
4: $A \wedge F \rightarrow P$
5: $A \wedge D \rightarrow P$
6: A
7: D

q: E p: D c: 5

cont:

1	2	3	4	5	6	7
1	2	2	1	1	0	0

inferida:

A	D	E	F	M	P
V	V	F	F	F	F

agenda: (vazia)

Forward Chaining: Algoritmo

Função *FORWARD_CHAINING(BC, q)*: **boolean**
- *cont* ← número de símbolos em cada premissa
- *inferida* ← inicializada com *falso* para todo símbolo
- *agenda* ← símbolos verdadeiros na *BC*
- **enquanto** *agenda não estiver vazia* **faça**
  - *p* ← retira o primeiro elemento de *agenda*
  - **se** $p = q$ **então**
    - **retorna** *verdadeiro*
  - **se** $inferida[p] = falso$ **então**
    - *inferida[p]* ← *verdadeiro*
    - **para cada** *cláusula c em que* $p \in PREMISSAS(c)$ **faça**
      - decremente $cont[c]$
      - **se** $cont[c] = 0$ **então**
        - Adicione a conclusão de *c* à *agenda*
- **retorna** *falso*;

BC:
1: $F \rightarrow E$
2: $P \wedge M \rightarrow F$
3: $D \wedge P \rightarrow M$
4: $A \wedge F \rightarrow P$
5: $A \wedge D \rightarrow P$
6: A
7: D

q: E p: D c: 3

cont:

1	2	3	4	5	6	7
1	2	2	1	1	0	0

inferida:

A	D	E	F	M	P
V	V	F	F	F	F

agenda: (vazia)

Forward Chaining: Algoritmo

Função *FORWARD_CHAINING(BC, q)*: **boolean**

```
cont ← número de símbolos em cada premissa
inferida ← inicializada com falso para todo símbolo
agenda ← símbolos verdadeiros na BC

enquanto agenda não estiver vazia faça
    p ← retira o primeiro elemento de agenda
    se p = q então
        retorna verdadeiro
    se inferida[p] = falso então
        inferida[p] ← verdadeiro
        para cada cláusula c em que p ∈ PREMISSAS(c) faça
            decremente cont[c]
            se cont[c] = 0 então
                Adicione a conclusão de c à agenda
retorna falso;
```

BC:

1: $F \rightarrow E$
2: $P \wedge M \rightarrow F$
3: $D \wedge P \rightarrow M$
4: $A \wedge F \rightarrow P$
5: $A \wedge D \rightarrow P$
6: A
7: D

q: E p: D c: 3

cont:

1	2	3	4	5	6	7
1	2	1	1	1	0	0

inferida:

A	D	E	F	M	P
V	V	F	F	F	F

agenda:

Forward Chaining: Algoritmo

Função *FORWARD_CHAINING(BC, q)*: **boolean**

$cont \leftarrow$ número de símbolos em cada premissa
$inferida \leftarrow$ inicializada com *falso* para todo símbolo
$agenda \leftarrow$ símbolos verdadeiros na BC

enquanto *agenda não estiver vazia* **faça**
  $p \leftarrow$ retira o primeiro elemento de *agenda*
  se $p = q$ **então**
    retorna *verdadeiro*
  se $inferida[p] = falso$ **então**
    $inferida[p] \leftarrow verdadeiro$
    para cada *cláusula c em que* $p \in PREMISSAS(c)$ **faça**
      decremente $cont[c]$
      se $cont[c] = 0$ **então**
        Adicione a conclusão de c à *agenda*

retorna *falso*;

BC:
1: $F \rightarrow E$
2: $P \wedge M \rightarrow F$
3: $D \wedge P \rightarrow M$
4: $A \wedge F \rightarrow P$
5: $A \wedge D \rightarrow P$
6: A
7: D

q: E p: D c: 3

cont:

1	2	3	4	5	6	7
1	2	1	1	1	0	0

inferida:

A	D	E	F	M	P
V	V	F	F	F	F

agenda: (vazia)

Lógica Proposicional e Inferência

Forward Chaining: Algoritmo

Função *FORWARD_CHAINING(BC, q)*: **boolean**

- *cont* ← número de símbolos em cada premissa
- *inferida* ← inicializada com *falso* para todo símbolo
- *agenda* ← símbolos verdadeiros na *BC*
- **enquanto** *agenda não estiver vazia* **faça**
  - *p* ← retira o primeiro elemento de *agenda*
  - **se** $p = q$ **então**
    - **retorna** *verdadeiro*
  - **se** $inferida[p] = falso$ **então**
    - *inferida[p]* ← *verdadeiro*
    - **para cada** *cláusula c em que* $p \in PREMISSAS(c)$ **faça**
      - decremente $cont[c]$
      - **se** $cont[c] = 0$ **então**
        - Adicione a conclusão de *c* à *agenda*
- **retorna** *falso*;

BC:

1:	$F \rightarrow E$
2:	$P \wedge M \rightarrow F$
3:	$D \wedge P \rightarrow M$
4:	$A \wedge F \rightarrow P$
5:	$A \wedge D \rightarrow P$
6:	A
7:	D

q: E p: D c: 5

cont:

1	2	3	4	5	6	7
1	2	1	1	1	0	0

inferida:

A	D	E	F	M	P
V	V	F	F	F	F

agenda: (vazia)

Forward Chaining: Algoritmo

Função *FORWARD_CHAINING(BC, q)*: **boolean**

- *cont* ← número de símbolos em cada premissa
- *inferida* ← inicializada com *falso* para todo símbolo
- *agenda* ← símbolos verdadeiros na *BC*
- **enquanto** *agenda não estiver vazia* **faça**
  - *p* ← retira o primeiro elemento de *agenda*
  - **se** $p = q$ **então**
    - **retorna** *verdadeiro*
  - **se** *inferida*[*p*] = *falso* **então**
    - *inferida[p]* ← *verdadeiro*
    - **para cada** *cláusula c em que* $p \in PREMISSAS(c)$ **faça**
      - decremente *cont*[*c*]
      - **se** *cont*[*c*] = 0 **então**
        - Adicione a conclusão de *c* à *agenda*
- **retorna** *falso*;

BC:
1: $F \rightarrow E$
2: $P \wedge M \rightarrow F$
3: $D \wedge P \rightarrow M$
4: $A \wedge F \rightarrow P$
5: $A \wedge D \rightarrow P$
6: A
7: D

q: E p: D c: 5

cont:

1	2	3	4	5	6	7
1	2	1	1	0	0	0

inferida:

A	D	E	F	M	P
V	V	F	F	F	F

agenda: (vazia)

Forward Chaining: Algoritmo

Função *FORWARD_CHAINING(BC, q)*: **boolean**

```
cont ← número de símbolos em cada premissa
inferida ← inicializada com falso para todo símbolo
agenda ← símbolos verdadeiros na BC

enquanto agenda não estiver vazia faça
    p ← retira o primeiro elemento de agenda
    se p = q então
        retorna verdadeiro
    se inferida[p] = falso então
        inferida[p] ← verdadeiro
        para cada cláusula c em que
        p ∈ PREMISSAS(c) faça
            decremente cont[c]
            se cont[c] = 0 então
                Adicione a conclusão de c à agenda
retorna falso;
```

BC:
1: $F \rightarrow E$
2: $P \wedge M \rightarrow F$
3: $D \wedge P \rightarrow M$
4: $A \wedge F \rightarrow P$
5: $A \wedge D \rightarrow P$
6: A
7: D

q: E p: D c: 5

cont:

1	2	3	4	5	6	7
1	2	1	1	0	0	0

inferida:

A	D	E	F	M	P
V	V	F	F	F	F

agenda: (vazia)

Lógica Proposicional e Inferência

Forward Chaining: Algoritmo

Função *FORWARD_CHAINING(BC, q)*: **boolean**

- *cont* ← número de símbolos em cada premissa
- *inferida* ← inicializada com *falso* para todo símbolo
- *agenda* ← símbolos verdadeiros na *BC*
- **enquanto** *agenda não estiver vazia* **faça**
  - *p* ← retira o primeiro elemento de *agenda*
  - **se** $p = q$ **então**
    - **retorna** *verdadeiro*
  - **se** $inferida[p] = falso$ **então**
    - *inferida[p]* ← *verdadeiro*
    - **para cada** *cláusula c em que* $p \in PREMISSAS(c)$ **faça**
      - decremente $cont[c]$
      - **se** $cont[c] = 0$ **então**
        - Adicione a conclusão de *c* à *agenda*
- **retorna** *falso*;

BC:

1: $F \rightarrow E$
2: $P \wedge M \rightarrow F$
3: $D \wedge P \rightarrow M$
4: $A \wedge F \rightarrow P$
5: $A \wedge D \rightarrow P$
6: A
7: D

q: E p: D c: 5

cont:

1	2	3	4	5	6	7
1	2	1	1	0	0	0

inferida:

A	D	E	F	M	P
V	V	F	F	F	F

agenda: P

Forward Chaining: Algoritmo

Função *FORWARD_CHAINING(BC, q)*: **boolean**
- *cont* ← número de símbolos em cada premissa
- *inferida* ← inicializada com *falso* para todo símbolo
- *agenda* ← símbolos verdadeiros na *BC*
- **enquanto** *agenda não estiver vazia* **faça**
  - *p* ← retira o primeiro elemento de *agenda*
  - **se** $p = q$ **então**
    - **retorna** *verdadeiro*
  - **se** $inferida[p] = falso$ **então**
    - *inferida[p]* ← *verdadeiro*
    - **para cada** cláusula *c* em que $p \in PREMISSAS(c)$ **faça**
      - decremente $cont[c]$
      - **se** $cont[c] = 0$ **então**
        - Adicione a conclusão de *c* à *agenda*
- **retorna** *falso*;

BC:
1: $F \rightarrow E$
2: $P \wedge M \rightarrow F$
3: $D \wedge P \rightarrow M$
4: $A \wedge F \rightarrow P$
5: $A \wedge D \rightarrow P$
6: A
7: D

q: E p: D c: 5

cont:

1	2	3	4	5	6	7
1	2	1	1	0	0	0

inferida:

A	*D*	*E*	*F*	*M*	*P*
V	*V*	*F*	*F*	*F*	*F*

agenda: *P*

Forward Chaining: Algoritmo

Função *FORWARD_CHAINING(BC, q)*: **boolean**

cont ← número de símbolos em cada premissa
inferida ← inicializada com *falso* para todo símbolo
agenda ← símbolos verdadeiros na *BC*

enquanto *agenda não estiver vazia* **faça**
  p ← retira o primeiro elemento de *agenda*
  se $p = q$ **então**
    retorna *verdadeiro*
  se $inferida[p] = falso$ **então**
    inferida[p] ← *verdadeiro*
    para cada *cláusula c em que* $p \in PREMISSAS(c)$ **faça**
      decremente $cont[c]$
      se $cont[c] = 0$ **então**
        Adicione a conclusão de c à *agenda*

retorna *falso*;

BC:
1: $F \rightarrow E$
2: $P \wedge M \rightarrow F$
3: $D \wedge P \rightarrow M$
4: $A \wedge F \rightarrow P$
5: $A \wedge D \rightarrow P$
6: A
7: D

q: E p: D c: 5

cont:

1	2	3	4	5	6	7
1	2	1	1	0	0	0

inferida:

A	D	E	F	M	P
V	V	F	F	F	F

agenda: P

Lógica Proposicional e Inferência

Forward Chaining: Algoritmo

Função *FORWARD_CHAINING(BC, q)*: **boolean**

- *cont* ← número de símbolos em cada premissa
- *inferida* ← inicializada com *falso* para todo símbolo
- *agenda* ← símbolos verdadeiros na *BC*
- **enquanto** *agenda não estiver vazia* **faça**
  - *p* ← retira o primeiro elemento de *agenda*
  - **se** $p = q$ **então**
    - **retorna** *verdadeiro*
  - **se** *inferida*$[p]$ = *falso* **então**
    - *inferida[p]* ← *verdadeiro*
    - **para cada** *cláusula c em que* $p \in PREMISSAS(c)$ **faça**
      - decremente *cont*$[c]$
      - **se** *cont*$[c] = 0$ **então**
        - Adicione a conclusão de *c* à *agenda*
- **retorna** *falso*;

BC:
1: $F \rightarrow E$
2: $P \wedge M \rightarrow F$
3: $D \wedge P \rightarrow M$
4: $A \wedge F \rightarrow P$
5: $A \wedge D \rightarrow P$
6: A
7: D

q: E p: P c: 5

cont:

1	2	3	4	5	6	7
1	2	1	1	0	0	0

inferida:

A	D	E	F	M	P
V	V	F	F	F	F

agenda: (vazia)

Forward Chaining: Algoritmo

Função *FORWARD_CHAINING(BC, q)*: **boolean**

- *cont* $\leftarrow$ número de símbolos em cada premissa
- *inferida* $\leftarrow$ inicializada com *falso* para todo símbolo
- *agenda* $\leftarrow$ símbolos verdadeiros na *BC*
- **enquanto** *agenda não estiver vazia* **faça**
  - $p \leftarrow$ retira o primeiro elemento de *agenda*
  - **se** $p = q$ **então**
    - **retorna** *verdadeiro*
  - **se** $inferida[p] = falso$ **então**
    - *inferida[p]* $\leftarrow$ *verdadeiro*
    - **para cada** *cláusula c em que* $p \in PREMISSAS(c)$ **faça**
      - decremente $cont[c]$
      - **se** $cont[c] = 0$ **então**
        - Adicione a conclusão de c à *agenda*
- **retorna** *falso*;

BC:

1: $F \rightarrow E$
2: $P \wedge M \rightarrow F$
3: $D \wedge P \rightarrow M$
4: $A \wedge F \rightarrow P$
5: $A \wedge D \rightarrow P$
6: A
7: D

q: E p: P c: 5

cont:

1	2	3	4	5	6	7
1	2	1	1	0	0	0

inferida:

A	*D*	*E*	*F*	*M*	*P*
V	*V*	*F*	*F*	*F*	*F*

agenda: (vazia)

Forward Chaining: Algoritmo

Função *FORWARD_CHAINING(BC, q)*: **boolean**
    cont ← número de símbolos em cada premissa
    inferida ← inicializada com *falso* para todo símbolo
    agenda ← símbolos verdadeiros na *BC*

    enquanto *agenda não estiver vazia* **faça**
        p ← retira o primeiro elemento de *agenda*
        se $p = q$ **então**
            retorna *verdadeiro*
        se $inferida[p] = falso$ **então**
            inferida[p] ← *verdadeiro*
            para cada *cláusula c em que*
            $p \in PREMISSAS(c)$ **faça**
                decremente $cont[c]$
                se $cont[c] = 0$ **então**
                    Adicione a conclusão de *c* à *agenda*
    retorna *falso*;

BC:
1: $F \rightarrow E$
2: $P \wedge M \rightarrow F$
3: $D \wedge P \rightarrow M$
4: $A \wedge F \rightarrow P$
5: $A \wedge D \rightarrow P$
6: A
7: D

q: E p: P c: 5

cont:

	1	2	3	4	5	6	7
cont:	1	2	1	1	0	0	0

inferida:

	A	D	E	F	M	P
inferida:	V	V	F	F	F	F

agenda: (vazia)

Forward Chaining: Algoritmo

Função *FORWARD_CHAINING(BC, q)*: **boolean**
  cont ← número de símbolos em cada premissa
  inferida ← inicializada com *falso* para todo símbolo
  agenda ← símbolos verdadeiros na *BC*

  enquanto *agenda não estiver vazia* **faça**
    p ← retira o primeiro elemento de *agenda*
    se $p = q$ **então**
      retorna *verdadeiro*
    se *inferida*$[p]$ = *falso* **então**
      inferida[p] ← verdadeiro
      para cada *cláusula c em que*
      $p \in PREMISSAS(c)$ **faça**
        decremente $cont[c]$
        se $cont[c] = 0$ **então**
          Adicione a conclusão de *c* à *agenda*
  retorna *falso*;

BC:
1: $F \rightarrow E$
2: $P \wedge M \rightarrow F$
3: $D \wedge P \rightarrow M$
4: $A \wedge F \rightarrow P$
5: $A \wedge D \rightarrow P$
6: A
7: D

q: E p: P c: 5

cont:

1	2	3	4	5	6	7
1	2	1	1	0	0	0

inferida:

A	*D*	*E*	*F*	*M*	*P*
V	*V*	*F*	*F*	*F*	*V*

agenda: (vazia)

Forward Chaining: Algoritmo

Função *FORWARD_CHAINING(BC, q)*: **boolean**

cont ← número de símbolos em cada premissa
inferida ← inicializada com *falso* para todo símbolo
agenda ← símbolos verdadeiros na *BC*

enquanto *agenda não estiver vazia* **faça**
  p ← retira o primeiro elemento de *agenda*
  se $p = q$ **então**
    retorna *verdadeiro*
  se $inferida[p] = falso$ **então**
    inferida[p] ← *verdadeiro*
    para cada cláusula *c* em que $p \in PREMISSAS(c)$ **faça**
      decremente $cont[c]$
      se $cont[c] = 0$ **então**
        Adicione a conclusão de *c* à *agenda*

retorna *falso*;

BC:
1: $F \rightarrow E$
2: $P \wedge M \rightarrow F$
3: $D \wedge P \rightarrow M$
4: $A \wedge F \rightarrow P$
5: $A \wedge D \rightarrow P$
6: A
7: D

q: E p: P c: 2

cont:

1	2	3	4	5	6	7
1	2	1	1	0	0	0

inferida:

A	D	E	F	M	P
V	V	F	F	F	V

agenda: (vazia)

Lógica Proposicional e Inferência

Forward Chaining: Algoritmo

Função *FORWARD_CHAINING(BC, q)*: **boolean**

- *cont* ← número de símbolos em cada premissa
- *inferida* ← inicializada com *falso* para todo símbolo
- *agenda* ← símbolos verdadeiros na *BC*
- **enquanto** *agenda não estiver vazia* **faça**
  - *p* ← retira o primeiro elemento de *agenda*
  - **se** $p = q$ **então**
    - **retorna** *verdadeiro*
  - **se** *inferida*$[p]$ = *falso* **então**
    - *inferida[p]* ← *verdadeiro*
    - **para cada** *cláusula c em que* $p \in PREMISSAS(c)$ **faça**
      - decremente $cont[c]$
      - **se** $cont[c] = 0$ **então**
        - Adicione a conclusão de *c* à *agenda*
- **retorna** *falso*;

BC:

1: $F \rightarrow E$
2: $P \wedge M \rightarrow F$
3: $D \wedge P \rightarrow M$
4: $A \wedge F \rightarrow P$
5: $A \wedge D \rightarrow P$
6: A
7: D

q: E p: P c: 2

cont:

1	2	3	4	5	6	7
1	1	1	1	0	0	0

inferida:

A	*D*	*E*	*F*	*M*	*P*
V	*V*	*F*	*F*	*F*	*V*

agenda:

Forward Chaining: Algoritmo

Função *FORWARD_CHAINING(BC, q)*: **boolean**
- *cont* ← número de símbolos em cada premissa
- *inferida* ← inicializada com *falso* para todo símbolo
- *agenda* ← símbolos verdadeiros na *BC*
- **enquanto** *agenda não estiver vazia* **faça**
  - *p* ← retira o primeiro elemento de *agenda*
  - **se** $p = q$ **então**
    - **retorna** *verdadeiro*
  - **se** $inferida[p] = falso$ **então**
    - *inferida[p]* ← *verdadeiro*
    - **para cada** *cláusula c em que* $p \in PREMISSAS(c)$ **faça**
      - decremente $cont[c]$
      - **se** $cont[c] = 0$ **então**
        - Adicione a conclusão de *c* à *agenda*
- **retorna** *falso*;

BC:
1: $F \rightarrow E$
2: $P \wedge M \rightarrow F$
3: $D \wedge P \rightarrow M$
4: $A \wedge F \rightarrow P$
5: $A \wedge D \rightarrow P$
6: A
7: D

q: E p: P c: 2

cont:

1	2	3	4	5	6	7
1	1	1	1	0	0	0

inferida:

A	*D*	*E*	*F*	*M*	*P*
V	*V*	*F*	*F*	*F*	*V*

agenda: (vazia)

Forward Chaining: Algoritmo

Função *FORWARD_CHAINING(BC, q)*: **boolean**
- *cont* ← número de símbolos em cada premissa
- *inferida* ← inicializada com *falso* para todo símbolo
- *agenda* ← símbolos verdadeiros na *BC*
- **enquanto** *agenda não estiver vazia* **faça**
  - *p* ← retira o primeiro elemento de *agenda*
  - **se** $p = q$ **então**
    - **retorna** *verdadeiro*
  - **se** $inferida[p] = falso$ **então**
    - *inferida[p]* ← *verdadeiro*
    - **para cada** cláusula *c* em que $p \in PREMISSAS(c)$ **faça**
      - decremente $cont[c]$
      - **se** $cont[c] = 0$ **então**
        - Adicione a conclusão de *c* à *agenda*
- **retorna** *falso*;

BC:
1: $F \rightarrow E$
2: $P \wedge M \rightarrow F$
3: $D \wedge P \rightarrow M$
4: $A \wedge F \rightarrow P$
5: $A \wedge D \rightarrow P$
6: A
7: D

q: E p: P c: 3

cont:

1	1	1	1	0	0	0
1	2	3	4	5	6	7

inferida:

V	V	F	F	F	V
A	D	E	F	M	P

agenda: (vazia)

Lógica Proposicional e Inferência

Forward Chaining: Algoritmo

Função *FORWARD_CHAINING(BC, q)*: **boolean**
- *cont* ← número de símbolos em cada premissa
- *inferida* ← inicializada com *falso* para todo símbolo
- *agenda* ← símbolos verdadeiros na *BC*
- **enquanto** *agenda não estiver vazia* **faça**
  - *p* ← retira o primeiro elemento de *agenda*
  - **se** $p = q$ **então**
    - **retorna** *verdadeiro*
  - **se** *inferida*$[p]$ = *falso* **então**
    - *inferida[p]* ← *verdadeiro*
    - **para cada** *cláusula c em que* $p \in PREMISSAS(c)$ **faça**
      - decremente *cont*$[c]$
      - **se** *cont*$[c] = 0$ **então**
        - Adicione a conclusão de *c* à *agenda*
- **retorna** *falso*;

BC:
1: $F \rightarrow E$
2: $P \wedge M \rightarrow F$
3: $D \wedge P \rightarrow M$
4: $A \wedge F \rightarrow P$
5: $A \wedge D \rightarrow P$
6: A
7: D

q: E p: P c: 3

cont:

1	2	3	4	5	6	7
1	1	0	1	0	0	0

inferida:

A	D	E	F	M	P
V	V	F	F	F	V

agenda:

Forward Chaining: Algoritmo

Função *FORWARD_CHAINING(BC, q)*: **boolean**
cont ← número de símbolos em cada premissa
inferida ← inicializada com *falso* para todo símbolo
agenda ← símbolos verdadeiros na *BC*

enquanto *agenda não estiver vazia* **faça**
  p ← retira o primeiro elemento de *agenda*
  se $p = q$ **então**
    retorna *verdadeiro*
  se $inferida[p] = falso$ **então**
    inferida[p] ← *verdadeiro*
    para cada *cláusula c em que* $p \in PREMISSAS(c)$ **faça**
      decremente $cont[c]$
      se $cont[c] = 0$ **então**
        Adicione a conclusão de *c* à *agenda*

retorna *falso*;

BC:
1: $F \to E$
2: $P \wedge M \to F$
3: $D \wedge P \to M$
4: $A \wedge F \to P$
5: $A \wedge D \to P$
6: A
7: D

q: E p: P c: 3

cont:

1	2	3	4	5	6	7
1	1	0	1	0	0	0

inferida:

A	D	E	F	M	P
V	V	F	F	F	V

agenda: (vazia)

Forward Chaining: Algoritmo

Função *FORWARD_CHAINING(BC, q)*: **boolean**
- *cont* ← número de símbolos em cada premissa
- *inferida* ← inicializada com *falso* para todo símbolo
- *agenda* ← símbolos verdadeiros na *BC*
- **enquanto** *agenda não estiver vazia* **faça**
  - *p* ← retira o primeiro elemento de *agenda*
  - **se** $p = q$ **então**
    - **retorna** *verdadeiro*
  - **se** $inferida[p] = falso$ **então**
    - *inferida[p]* ← *verdadeiro*
    - **para cada** *cláusula c em que* $p \in PREMISSAS(c)$ **faça**
      - decremente $cont[c]$
      - **se** $cont[c] = 0$ **então**
        - Adicione a conclusão de *c* à *agenda*
- **retorna** *falso*;

BC:
1: $F \rightarrow E$
2: $P \wedge M \rightarrow F$
3: $D \wedge P \rightarrow M$
4: $A \wedge F \rightarrow P$
5: $A \wedge D \rightarrow P$
6: A
7: D

q: E p: P c: 3

cont:

1	2	3	4	5	6	7
1	1	0	1	0	0	0

inferida:

A	D	E	F	M	P
V	V	F	F	F	V

agenda: $\boxed{M}$

Forward Chaining: Algoritmo

Função *FORWARD_CHAINING(BC, q)*: **boolean**
- *cont* ← número de símbolos em cada premissa
- *inferida* ← inicializada com *falso* para todo símbolo
- *agenda* ← símbolos verdadeiros na *BC*
- **enquanto** *agenda não estiver vazia* **faça**
  - *p* ← retira o primeiro elemento de *agenda*
  - **se** $p = q$ **então**
    - **retorna** *verdadeiro*
  - **se** $inferida[p] = falso$ **então**
    - *inferida[p]* ← *verdadeiro*
    - **para cada** cláusula *c* em que $p \in PREMISSAS(c)$ **faça**
      - decremente $cont[c]$
      - **se** $cont[c] = 0$ **então**
        - Adicione a conclusão de *c* à *agenda*
- **retorna** *falso*;

BC:

1:	$F \rightarrow E$
2:	$P \wedge M \rightarrow F$
3:	$D \wedge P \rightarrow M$
4:	$A \wedge F \rightarrow P$
5:	$A \wedge D \rightarrow P$
6:	A
7:	D

q: E p: P c: 3

cont:

1	2	3	4	5	6	7
1	1	0	1	0	0	0

inferida:

A	D	E	F	M	P
V	V	F	F	F	V

agenda: M

Forward Chaining: Algoritmo

Função *FORWARD_CHAINING(BC, q)*: **boolean**

cont ← número de símbolos em cada premissa
inferida ← inicializada com *falso* para todo símbolo
agenda ← símbolos verdadeiros na *BC*

enquanto *agenda não estiver vazia* **faça**
- *p* ← retira o primeiro elemento de *agenda*
- **se** $p = q$ **então**
  - **retorna** *verdadeiro*
- **se** $inferida[p] = falso$ **então**
  - *inferida[p]* ← *verdadeiro*
  - **para cada** *cláusula c em que* $p \in PREMISSAS(c)$ **faça**
    - decremente $cont[c]$
    - **se** $cont[c] = 0$ **então**
      - Adicione a conclusão de *c* à *agenda*

retorna *falso*;

BC:
1: $F \rightarrow E$
2: $P \wedge M \rightarrow F$
3: $D \wedge P \rightarrow M$
4: $A \wedge F \rightarrow P$
5: $A \wedge D \rightarrow P$
6: A
7: D

q: E p: P c: 3

cont:

1	2	3	4	5	6	7
1	1	0	1	0	0	0

inferida:

A	D	E	F	M	P
V	V	F	F	F	V

agenda: M

Lógica Proposicional e Inferência

Forward Chaining: Algoritmo

Função *FORWARD_CHAINING(BC, q)*: **boolean**

cont ← número de símbolos em cada premissa
inferida ← inicializada com *falso* para todo símbolo
agenda ← símbolos verdadeiros na *BC*

enquanto *agenda não estiver vazia* **faça**
- p ← retira o primeiro elemento de *agenda*
- **se** $p = q$ **então**
  - **retorna** *verdadeiro*
- **se** *inferida*$[p]$ = *falso* **então**
  - *inferida[p]* ← *verdadeiro*
  - **para cada** *cláusula c em que* $p \in PREMISSAS(c)$ **faça**
    - decremente *cont*$[c]$
    - **se** *cont*$[c] = 0$ **então**
      - Adicione a conclusão de c à *agenda*

retorna *falso*;

BC:
1: $F \rightarrow E$
2: $P \wedge M \rightarrow F$
3: $D \wedge P \rightarrow M$
4: $A \wedge F \rightarrow P$
5: $A \wedge D \rightarrow P$
6: A
7: D

q: E p: M c: 3

cont:

1	2	3	4	5	6	7
1	1	0	1	0	0	0

inferida:

A	D	E	F	M	P
V	V	F	F	F	V

agenda: (vazia)

Forward Chaining: Algoritmo

Função *FORWARD_CHAINING(BC, q)*: **boolean**
- *cont* ← número de símbolos em cada premissa
- *inferida* ← inicializada com *falso* para todo símbolo
- *agenda* ← símbolos verdadeiros na *BC*
- **enquanto** *agenda não estiver vazia* **faça**
  - *p* ← retira o primeiro elemento de *agenda*
  - **se** $p = q$ **então**
    - **retorna** *verdadeiro*
  - **se** $inferida[p] = falso$ **então**
    - *inferida[p]* ← *verdadeiro*
    - **para cada** *cláusula c em que* $p \in PREMISSAS(c)$ **faça**
      - decremente $cont[c]$
      - **se** $cont[c] = 0$ **então**
        - Adicione a conclusão de *c* à *agenda*
- **retorna** *falso*;

BC:

1:	$F \rightarrow E$
2:	$P \wedge M \rightarrow F$
3:	$D \wedge P \rightarrow M$
4:	$A \wedge F \rightarrow P$
5:	$A \wedge D \rightarrow P$
6:	A
7:	D

q: E p: M c: 3

cont:

1	2	3	4	5	6	7
1	1	0	1	0	0	0

inferida:

A	D	E	F	M	P
V	V	F	F	F	V

agenda: (vazia)

Lógica Proposicional e Inferência

Forward Chaining: Algoritmo

Função *FORWARD_CHAINING(BC, q)*: **boolean**

```
cont ← número de símbolos em cada premissa
inferida ← inicializada com falso para todo símbolo
agenda ← símbolos verdadeiros na BC

enquanto agenda não estiver vazia faça
    p ← retira o primeiro elemento de agenda
    se p = q então
        retorna verdadeiro
    se inferida[p] = falso então
        inferida[p] ← verdadeiro
        para cada cláusula c em que
        p ∈ PREMISSAS(c) faça
            decremente cont[c]
            se cont[c] = 0 então
                Adicione a conclusão de c à agenda
retorna falso;
```

BC:

1: $F \rightarrow E$
2: $P \wedge M \rightarrow F$
3: $D \wedge P \rightarrow M$
4: $A \wedge F \rightarrow P$
5: $A \wedge D \rightarrow P$
6: A
7: D

q: E p: M c: 3

cont:

1	2	3	4	5	6	7
1	1	0	1	0	0	0

inferida:

A	D	E	F	M	P
V	V	F	F	F	V

agenda:

Forward Chaining: Algoritmo

Função *FORWARD_CHAINING(BC, q)*: **boolean**
  cont ← número de símbolos em cada premissa
  inferida ← inicializada com *falso* para todo símbolo
  agenda ← símbolos verdadeiros na *BC*

  enquanto *agenda não estiver vazia* **faça**
    p ← retira o primeiro elemento de *agenda*
    se $p = q$ **então**
      retorna *verdadeiro*
    se *inferida*[*p*] = *falso* **então**
      inferida[p] ← verdadeiro
      para cada *cláusula c em que*
      $p \in PREMISSAS(c)$ **faça**
        decremente *cont*[*c*]
        se *cont*[*c*] = 0 **então**
          Adicione a conclusão de *c* à *agenda*
  retorna *falso*;

BC:
1: $F \rightarrow E$
2: $P \wedge M \rightarrow F$
3: $D \wedge P \rightarrow M$
4: $A \wedge F \rightarrow P$
5: $A \wedge D \rightarrow P$
6: A
7: D

q: E p: M c: 3

cont:

1	2	3	4	5	6	7
1	1	0	1	0	0	0

inferida:

A	D	E	F	M	P
V	V	F	F	V	V

agenda: (vazia)

Forward Chaining: Algoritmo

Função *FORWARD_CHAINING(BC, q)*: **boolean**
  cont ← número de símbolos em cada premissa
  inferida ← inicializada com *falso* para todo símbolo
  agenda ← símbolos verdadeiros na *BC*

  enquanto *agenda não estiver vazia* **faça**
    p ← retira o primeiro elemento de *agenda*
    se $p = q$ **então**
      retorna *verdadeiro*
    se *inferida*[*p*] = *falso* **então**
      inferida[p] ← *verdadeiro*
      para cada *cláusula c em que* $p \in PREMISSAS(c)$ **faça**
        decremente *cont*[*c*]
        se *cont*[*c*] = 0 **então**
          Adicione a conclusão de *c* à *agenda*
  retorna *falso*;

BC:
1: $F \rightarrow E$
2: $P \wedge M \rightarrow F$
3: $D \wedge P \rightarrow M$
4: $A \wedge F \rightarrow P$
5: $A \wedge D \rightarrow P$
6: A
7: D

q: E p: M c: 2

cont:

1	2	3	4	5	6	7
1	1	0	1	0	0	0

inferida:

A	D	E	F	M	P
V	V	F	F	V	V

agenda: (vazia)

Forward Chaining: Algoritmo

Função *FORWARD_CHAINING(BC, q)*: **boolean**
- *cont* ← número de símbolos em cada premissa
- *inferida* ← inicializada com *falso* para todo símbolo
- *agenda* ← símbolos verdadeiros na *BC*
- **enquanto** *agenda não estiver vazia* **faça**
  - *p* ← retira o primeiro elemento de *agenda*
  - **se** $p = q$ **então**
    - **retorna** *verdadeiro*
  - **se** $inferida[p] = falso$ **então**
    - *inferida[p]* ← *verdadeiro*
    - **para cada** *cláusula c em que* $p \in PREMISSAS(c)$ **faça**
      - decremente $cont[c]$
      - **se** $cont[c] = 0$ **então**
        - Adicione a conclusão de *c* à *agenda*
- **retorna** *falso*;

BC:
1: $F \rightarrow E$
2: $P \wedge M \rightarrow F$
3: $D \wedge P \rightarrow M$
4: $A \wedge F \rightarrow P$
5: $A \wedge D \rightarrow P$
6: A
7: D

q: E p: M c: 2

cont:

1	2	3	4	5	6	7
1	0	0	1	0	0	0

inferida:

A	D	E	F	M	P
V	V	F	F	V	V

agenda: []

Lógica Proposicional e Inferência

Forward Chaining: Algoritmo

Função *FORWARD_CHAINING(BC, q)*: **boolean**
    cont ← número de símbolos em cada premissa
    inferida ← inicializada com *falso* para todo símbolo
    agenda ← símbolos verdadeiros na *BC*

    enquanto *agenda não estiver vazia* **faça**
        p ← retira o primeiro elemento de *agenda*
        se $p = q$ **então**
            retorna *verdadeiro*
        se $inferida[p] = falso$ **então**
            inferida[p] ← *verdadeiro*
            para cada *cláusula c em que* $p \in PREMISSAS(c)$ **faça**
                decremente $cont[c]$
                se $cont[c] = 0$ **então**
                    Adicione a conclusão de *c* à *agenda*
    retorna *falso*;

BC:
1: $F \to E$
2: $P \wedge M \to F$
3: $D \wedge P \to M$
4: $A \wedge F \to P$
5: $A \wedge D \to P$
6: A
7: D

q: E p: M c: 2

cont:

1	2	3	4	5	6	7
1	0	0	1	0	0	0

inferida:

A	*D*	*E*	*F*	*M*	*P*
V	*V*	*F*	*F*	*V*	*V*

agenda: (vazia)

Forward Chaining: Algoritmo

Função *FORWARD_CHAINING(BC, q)*: **boolean**
  cont ← número de símbolos em cada premissa
  inferida ← inicializada com *falso* para todo símbolo
  agenda ← símbolos verdadeiros na *BC*

  enquanto *agenda não estiver vazia* **faça**
    p ← retira o primeiro elemento de *agenda*
    se $p = q$ **então**
      retorna *verdadeiro*
    se $inferida[p] = falso$ **então**
      inferida[p] ← *verdadeiro*
      para cada *cláusula c em que* $p \in PREMISSAS(c)$ **faça**
        decremente $cont[c]$
        se $cont[c] = 0$ **então**
          Adicione a conclusão de *c* à *agenda*
  retorna *falso*;

BC:

1:	$F \to E$
2:	$P \wedge M \to F$
3:	$D \wedge P \to M$
4:	$A \wedge F \to P$
5:	$A \wedge D \to P$
6:	A
7:	D

q: E p: M c: 2

cont:

1	2	3	4	5	6	7
1	0	0	1	0	0	0

inferida:

A	D	E	F	M	P
V	V	F	F	V	V

agenda: F

Lógica Proposicional e Inferência

Forward Chaining: Algoritmo

Função *FORWARD_CHAINING(BC, q)*: **boolean**

- *cont* ← número de símbolos em cada premissa
- *inferida* ← inicializada com *falso* para todo símbolo
- *agenda* ← símbolos verdadeiros na *BC*
- **enquanto** *agenda não estiver vazia* **faça**
  - p ← retira o primeiro elemento de *agenda*
  - **se** $p = q$ **então**
    - **retorna** *verdadeiro*
  - **se** $inferida[p] = falso$ **então**
    - *inferida[p]* ← *verdadeiro*
    - **para cada** *cláusula c em que* $p \in PREMISSAS(c)$ **faça**
      - decremente $cont[c]$
      - **se** $cont[c] = 0$ **então**
        - Adicione a conclusão de c à *agenda*
- **retorna** *falso*;

BC:

1: $F \rightarrow E$
2: $P \wedge M \rightarrow F$
3: $D \wedge P \rightarrow M$
4: $A \wedge F \rightarrow P$
5: $A \wedge D \rightarrow P$
6: A
7: D

q: E p: M c: 2

cont:

1	2	3	4	5	6	7
1	0	0	1	0	0	0

inferida:

A	D	E	F	M	P
V	V	F	F	V	V

agenda: F

Lógica Proposicional e Inferência

Forward Chaining: Algoritmo

Função *FORWARD_CHAINING(BC, q)*: **boolean**
- *cont* ← número de símbolos em cada premissa
- *inferida* ← inicializada com *falso* para todo símbolo
- *agenda* ← símbolos verdadeiros na *BC*
- **enquanto** *agenda não estiver vazia* **faça**
  - *p* ← retira o primeiro elemento de *agenda*
  - **se** $p = q$ **então**
    - **retorna** *verdadeiro*
  - **se** $inferida[p] = falso$ **então**
    - *inferida[p]* ← *verdadeiro*
    - **para cada** *cláusula c em que* $p \in PREMISSAS(c)$ **faça**
      - decremente $cont[c]$
      - **se** $cont[c] = 0$ **então**
        - Adicione a conclusão de *c* à *agenda*
- **retorna** *falso*;

BC:
1: $F \rightarrow E$
2: $P \wedge M \rightarrow F$
3: $D \wedge P \rightarrow M$
4: $A \wedge F \rightarrow P$
5: $A \wedge D \rightarrow P$
6: A
7: D

q: E p: M c: 2

cont:

1	2	3	4	5	6	7
1	0	0	1	0	0	0

inferida:

A	D	E	F	M	P
V	V	F	F	V	V

agenda: F

Lógica Proposicional e Inferência

Forward Chaining: Algoritmo

Função *FORWARD_CHAINING(BC, q)*: **boolean**

> *cont* ← número de símbolos em cada premissa
> *inferida* ← inicializada com *falso* para todo símbolo
> *agenda* ← símbolos verdadeiros na *BC*
>
> **enquanto** *agenda não estiver vazia* **faça**
> > *p* ← retira o primeiro elemento de *agenda*
> > **se** $p = q$ **então**
> > > **retorna** *verdadeiro*
> >
> > **se** *inferida*$[p]$ $=$ *falso* **então**
> > > *inferida[p]* ← *verdadeiro*
> > > **para cada** *cláusula c em que* $p \in PREMISSAS(c)$ **faça**
> > > > decremente *cont*$[c]$
> > > > **se** *cont*$[c] = 0$ **então**
> > > > > Adicione a conclusão de *c* à *agenda*
>
> **retorna** *falso*;

BC:
1: $F \rightarrow E$
2: $P \wedge M \rightarrow F$
3: $D \wedge P \rightarrow M$
4: $A \wedge F \rightarrow P$
5: $A \wedge D \rightarrow P$
6: A
7: D

q: E p: F c: 2

cont:

1	2	3	4	5	6	7
1	0	0	1	0	0	0

inferida:

A	D	E	F	M	P
V	V	F	F	V	V

agenda:

Forward Chaining: Algoritmo

Função *FORWARD_CHAINING(BC, q)*: **boolean**
- *cont* ← número de símbolos em cada premissa
- *inferida* ← inicializada com *falso* para todo símbolo
- *agenda* ← símbolos verdadeiros na *BC*
- **enquanto** *agenda não estiver vazia* **faça**
  - *p* ← retira o primeiro elemento de *agenda*
  - **se** $p = q$ **então**
    - **retorna** *verdadeiro*
  - **se** $inferida[p] = falso$ **então**
    - *inferida[p]* ← *verdadeiro*
    - **para cada** *cláusula c em que* $p \in PREMISSAS(c)$ **faça**
      - decremente $cont[c]$
      - **se** $cont[c] = 0$ **então**
        - Adicione a conclusão de *c* à *agenda*
- **retorna** *falso*;

BC:
1: $F \to E$
2: $P \wedge M \to F$
3: $D \wedge P \to M$
4: $A \wedge F \to P$
5: $A \wedge D \to P$
6: A
7: D

q: E p: F c: 2

cont:

1	2	3	4	5	6	7
1	0	0	1	0	0	0

inferida:

A	D	E	F	M	P
V	V	F	F	V	V

agenda:

Lógica Proposicional e Inferência

Forward Chaining: Algoritmo

Função *FORWARD_CHAINING(BC, q)*: **boolean**

cont ← número de símbolos em cada premissa
inferida ← inicializada com *falso* para todo símbolo
agenda ← símbolos verdadeiros na *BC*

enquanto *agenda não estiver vazia* **faça**
  p ← retira o primeiro elemento de *agenda*
  se $p = q$ **então**
    retorna *verdadeiro*
  se $inferida[p] = falso$ **então**
    $inferida[p] \leftarrow verdadeiro$
    para cada *cláusula c em que* $p \in PREMISSAS(c)$ **faça**
      decremente $cont[c]$
      se $cont[c] = 0$ **então**
        Adicione a conclusão de *c* à *agenda*

retorna *falso*;

BC:
1: $F \rightarrow E$
2: $P \wedge M \rightarrow F$
3: $D \wedge P \rightarrow M$
4: $A \wedge F \rightarrow P$
5: $A \wedge D \rightarrow P$
6: A
7: D

q: E p: F c: 2

cont:

1	2	3	4	5	6	7
1	0	0	1	0	0	0

inferida:

A	D	E	F	M	P
V	V	F	F	V	V

agenda: (vazia)

Forward Chaining: Algoritmo

Função *FORWARD_CHAINING(BC, q)*: **boolean**
  cont ← número de símbolos em cada premissa
  inferida ← inicializada com *falso* para todo símbolo
  agenda ← símbolos verdadeiros na *BC*

  enquanto *agenda não estiver vazia* **faça**
    p ← retira o primeiro elemento de *agenda*
    se $p = q$ **então**
      retorna *verdadeiro*
    se *inferida*[*p*] = *falso* **então**
      inferida[p] ← verdadeiro
      para cada *cláusula c em que* $p \in PREMISSAS(c)$ **faça**
        decremente *cont*[*c*]
        se *cont*[*c*] = 0 **então**
          Adicione a conclusão de *c* à *agenda*
  retorna *falso*;

BC:
1: $F \rightarrow E$
2: $P \wedge M \rightarrow F$
3: $D \wedge P \rightarrow M$
4: $A \wedge F \rightarrow P$
5: $A \wedge D \rightarrow P$
6: A
7: D

q: E p: F c: 2

cont:

1	2	3	4	5	6	7
1	0	0	1	0	0	0

inferida:

A	D	E	F	M	P
V	V	F	V	V	V

agenda:

Forward Chaining: Algoritmo

Função *FORWARD_CHAINING(BC, q)*: **boolean**
- *cont* ← número de símbolos em cada premissa
- *inferida* ← inicializada com *falso* para todo símbolo
- *agenda* ← símbolos verdadeiros na *BC*
- **enquanto** *agenda não estiver vazia* **faça**
  - *p* ← retira o primeiro elemento de *agenda*
  - **se** $p = q$ **então**
    - **retorna** *verdadeiro*
  - **se** $inferida[p] = falso$ **então**
    - *inferida[p]* ← *verdadeiro*
    - **para cada** cláusula *c* em que $p \in PREMISSAS(c)$ **faça**
      - decremente $cont[c]$
      - **se** $cont[c] = 0$ **então**
        - Adicione a conclusão de *c* à *agenda*
- **retorna** *falso*;

BC:

1: $F \to E$
2: $P \wedge M \to F$
3: $D \wedge P \to M$
4: $A \wedge F \to P$
5: $A \wedge D \to P$
6: A
7: D

q: E p: F c: 1

cont:

1	2	3	4	5	6	7
1	0	0	1	0	0	0

inferida:

A	D	E	F	M	P
V	V	F	V	V	V

agenda: (vazia)

Forward Chaining: Algoritmo

Função *FORWARD_CHAINING(BC, q)*: **boolean**
- *cont* $\leftarrow$ número de símbolos em cada premissa
- *inferida* $\leftarrow$ inicializada com *falso* para todo símbolo
- *agenda* $\leftarrow$ símbolos verdadeiros na *BC*
- **enquanto** *agenda não estiver vazia* **faça**
  - $p \leftarrow$ retira o primeiro elemento de *agenda*
  - **se** $p = q$ **então**
    - **retorna** *verdadeiro*
  - **se** $inferida[p] = falso$ **então**
    - *inferida[p]* $\leftarrow$ *verdadeiro*
    - **para cada** *cláusula c em que* $p \in PREMISSAS(c)$ **faça**
      - decremente $cont[c]$
      - **se** $cont[c] = 0$ **então**
        - Adicione a conclusão de *c* à *agenda*
- **retorna** *falso*;

BC:
1: $F \rightarrow E$
2: $P \wedge M \rightarrow F$
3: $D \wedge P \rightarrow M$
4: $A \wedge F \rightarrow P$
5: $A \wedge D \rightarrow P$
6: A
7: D

q: E p: F c: 1

cont:

1	2	3	4	5	6	7
0	0	0	1	0	0	0

inferida:

A	D	E	F	M	P
V	V	F	V	V	V

agenda: (vazia)

Forward Chaining: Algoritmo

Função *FORWARD_CHAINING(BC, q)*: **boolean**
  cont ← número de símbolos em cada premissa
  inferida ← inicializada com *falso* para todo símbolo
  agenda ← símbolos verdadeiros na *BC*

  enquanto *agenda não estiver vazia* **faça**
    p ← retira o primeiro elemento de *agenda*
    se $p = q$ **então**
      retorna *verdadeiro*
    se $inferida[p] = falso$ **então**
      inferida[p] ← *verdadeiro*
      para cada *cláusula c em que*
      $p \in PREMISSAS(c)$ **faça**
        decremente $cont[c]$
        se $cont[c] = 0$ **então**
          Adicione a conclusão de *c* à *agenda*
  retorna *falso*;

BC:
1: $F \rightarrow E$
2: $P \wedge M \rightarrow F$
3: $D \wedge P \rightarrow M$
4: $A \wedge F \rightarrow P$
5: $A \wedge D \rightarrow P$
6: A
7: D

q: E p: F c: 1

cont:

1	2	3	4	5	6	7
0	0	0	1	0	0	0

inferida:

A	D	E	F	M	P
V	V	F	V	V	V

agenda: (vazia)

Forward Chaining: Algoritmo

Função *FORWARD_CHAINING(BC, q)*: **boolean**
- *cont* ← número de símbolos em cada premissa
- *inferida* ← inicializada com *falso* para todo símbolo
- *agenda* ← símbolos verdadeiros na *BC*
- **enquanto** *agenda não estiver vazia* **faça**
  - *p* ← retira o primeiro elemento de *agenda*
  - **se** $p = q$ **então**
    - **retorna** *verdadeiro*
  - **se** $inferida[p] = falso$ **então**
    - *inferida[p]* ← *verdadeiro*
    - **para cada** *cláusula c em que* $p \in PREMISSAS(c)$ **faça**
      - decremente $cont[c]$
      - **se** $cont[c] = 0$ **então**
        - Adicione a conclusão de *c* à *agenda*
- **retorna** *falso*;

BC:

1:	$F \rightarrow E$
2:	$P \wedge M \rightarrow F$
3:	$D \wedge P \rightarrow M$
4:	$A \wedge F \rightarrow P$
5:	$A \wedge D \rightarrow P$
6:	A
7:	D

q: E p: F c: 1

cont:

1	2	3	4	5	6	7
0	0	0	1	0	0	0

inferida:

A	D	E	F	M	P
V	V	F	V	V	V

agenda: E

Forward Chaining: Algoritmo

Função *FORWARD_CHAINING(BC, q)*: **boolean**

```
cont ← número de símbolos em cada premissa
inferida ← inicializada com falso para todo símbolo
agenda ← símbolos verdadeiros na BC

enquanto agenda não estiver vazia faça
    p ← retira o primeiro elemento de agenda
    se p = q então
        retorna verdadeiro
    se inferida[p] = falso então
        inferida[p] ← verdadeiro
        para cada cláusula c em que p ∈ PREMISSAS(c) faça
            decremente cont[c]
            se cont[c] = 0 então
                Adicione a conclusão de c à agenda
retorna falso;
```

BC:

1:	$F \rightarrow E$
2:	$P \wedge M \rightarrow F$
3:	$D \wedge P \rightarrow M$
4:	$A \wedge F \rightarrow P$
5:	$A \wedge D \rightarrow P$
6:	A
7:	D

q: E p: F c: 4

cont:

1	2	3	4	5	6	7
0	0	0	1	0	0	0

inferida:

A	D	E	F	M	P
V	V	F	V	V	V

agenda: E

Forward Chaining: Algoritmo

Função *FORWARD_CHAINING(BC, q)*: **boolean**

```
cont ← número de símbolos em cada premissa
inferida ← inicializada com falso para todo símbolo
agenda ← símbolos verdadeiros na BC

enquanto agenda não estiver vazia faça
    p ← retira o primeiro elemento de agenda
    se p = q então
        retorna verdadeiro
    se inferida[p] = falso então
        inferida[p] ← verdadeiro
        para cada cláusula c em que
        p ∈ PREMISSAS(c) faça
            decremente cont[c]
            se cont[c] = 0 então
                Adicione a conclusão de c à agenda
retorna falso;
```

BC:
1: $F \to E$
2: $P \wedge M \to F$
3: $D \wedge P \to M$
4: $A \wedge F \to P$
5: $A \wedge D \to P$
6: A
7: D

q: E p: F c: 4

cont:

1	2	3	4	5	6	7
0	0	0	0	0	0	0

inferida:

A	D	E	F	M	P
V	V	F	V	V	V

agenda: E

Forward Chaining: Algoritmo

Função *FORWARD_CHAINING(BC, q)*: **boolean**
  cont ← número de símbolos em cada premissa
  inferida ← inicializada com *falso* para todo símbolo
  agenda ← símbolos verdadeiros na *BC*

  enquanto *agenda não estiver vazia* **faça**
    p ← retira o primeiro elemento de *agenda*
    se $p = q$ **então**
      retorna *verdadeiro*
    se $inferida[p] = falso$ **então**
      inferida[p] ← *verdadeiro*
      para cada *cláusula c em que* $p \in PREMISSAS(c)$ **faça**
        decremente $cont[c]$
        se $cont[c] = 0$ **então**
          Adicione a conclusão de *c* à *agenda*
  retorna *falso*;

BC:
1: $F \rightarrow E$
2: $P \wedge M \rightarrow F$
3: $D \wedge P \rightarrow M$
4: $A \wedge F \rightarrow P$
5: $A \wedge D \rightarrow P$
6: A
7: D

q: E p: F c: 4

cont:

1	2	3	4	5	6	7
0	0	0	0	0	0	0

inferida:

A	D	E	F	M	P
V	V	F	V	V	V

agenda: E

Lógica Proposicional e Inferência

Forward Chaining: Algoritmo

Função *FORWARD_CHAINING(BC, q)*: **boolean**

- *cont* ← número de símbolos em cada premissa
- *inferida* ← inicializada com *falso* para todo símbolo
- *agenda* ← símbolos verdadeiros na *BC*
- **enquanto** *agenda não estiver vazia* **faça**
  - *p* ← retira o primeiro elemento de *agenda*
  - **se** $p = q$ **então**
    - **retorna** *verdadeiro*
  - **se** $inferida[p] = falso$ **então**
    - *inferida[p]* ← *verdadeiro*
    - **para cada** *cláusula c em que* $p \in PREMISSAS(c)$ **faça**
      - decremente $cont[c]$
      - **se** $cont[c] = 0$ **então**
        - Adicione a conclusão de *c* à *agenda*
- **retorna** *falso*;

BC:

1: $F \rightarrow E$
2: $P \wedge M \rightarrow F$
3: $D \wedge P \rightarrow M$
4: $A \wedge F \rightarrow P$
5: $A \wedge D \rightarrow P$
6: A
7: D

q: E p: F c: 4

cont:

1	2	3	4	5	6	7
0	0	0	0	0	0	0

inferida:

A	D	E	F	M	P
V	V	F	V	V	V

agenda:

E	P

Forward Chaining: Algoritmo

Função *FORWARD_CHAINING(BC, q)*: **boolean**
  cont ← número de símbolos em cada premissa
  inferida ← inicializada com *falso* para todo símbolo
  agenda ← símbolos verdadeiros na *BC*

  enquanto *agenda não estiver vazia* **faça**
    p ← retira o primeiro elemento de *agenda*
    se $p = q$ **então**
      retorna *verdadeiro*
    se *inferida*[*p*] = *falso* **então**
      inferida[p] ← *verdadeiro*
      para cada *cláusula c em que* $p \in PREMISSAS(c)$ **faça**
        decremente *cont*[*c*]
        se *cont*[*c*] = 0 **então**
          Adicione a conclusão de *c* à *agenda*
  retorna *falso*;

BC:
1: $F \to E$
2: $P \wedge M \to F$
3: $D \wedge P \to M$
4: $A \wedge F \to P$
5: $A \wedge D \to P$
6: A
7: D

q: E p: F c: 4

cont:

1	2	3	4	5	6	7
0	0	0	0	0	0	0

inferida:

A	D	E	F	M	P
V	V	F	V	V	V

agenda:

E	P

Forward Chaining: Algoritmo

Função *FORWARD_CHAINING(BC, q)*: **boolean**

```
cont ← número de símbolos em cada premissa
inferida ← inicializada com falso para todo símbolo
agenda ← símbolos verdadeiros na BC

enquanto agenda não estiver vazia faça
    p ← retira o primeiro elemento de agenda
    se p = q então
        retorna verdadeiro
    se inferida[p] = falso então
        inferida[p] ← verdadeiro
        para cada cláusula c em que p ∈ PREMISSAS(c) faça
            decremente cont[c]
            se cont[c] = 0 então
                Adicione a conclusão de c à agenda
retorna falso;
```

BC:
1: $F \rightarrow E$
2: $P \wedge M \rightarrow F$
3: $D \wedge P \rightarrow M$
4: $A \wedge F \rightarrow P$
5: $A \wedge D \rightarrow P$
6: A
7: D

q: E p: F c: 4

cont:

1	2	3	4	5	6	7
0	0	0	0	0	0	0

inferida:

A	D	E	F	M	P
V	V	F	V	V	V

agenda:

E	P

Forward Chaining: Algoritmo

Função *FORWARD_CHAINING(BC, q)*: **boolean**
- *cont* $\leftarrow$ número de símbolos em cada premissa
- *inferida* $\leftarrow$ inicializada com *falso* para todo símbolo
- *agenda* $\leftarrow$ símbolos verdadeiros na *BC*
- **enquanto** *agenda não estiver vazia* **faça**
  - *p* $\leftarrow$ retira o primeiro elemento de *agenda*
  - **se** $p = q$ **então**
    - **retorna** *verdadeiro*
  - **se** *inferida*$[p]$ = *falso* **então**
    - *inferida[p]* $\leftarrow$ *verdadeiro*
    - **para cada** *cláusula c em que* $p \in PREMISSAS(c)$ **faça**
      - decremente *cont*$[c]$
      - **se** *cont*$[c] = 0$ **então**
        - Adicione a conclusão de *c* à *agenda*
- **retorna** *falso*;

BC:
1: $F \rightarrow E$
2: $P \land M \rightarrow F$
3: $D \land P \rightarrow M$
4: $A \land F \rightarrow P$
5: $A \land D \rightarrow P$
6: A
7: D

q: E p: E c: 4

cont:

1	2	3	4	5	6	7
0	0	0	0	0	0	0

inferida:

A	D	E	F	M	P
V	V	F	V	V	V

agenda:

P

Forward Chaining: Algoritmo

Função *FORWARD_CHAINING(BC, q)*: **boolean**
- *cont* ← número de símbolos em cada premissa
- *inferida* ← inicializada com *falso* para todo símbolo
- *agenda* ← símbolos verdadeiros na *BC*
- **enquanto** *agenda não estiver vazia* **faça**
  - *p* ← retira o primeiro elemento de *agenda*
  - **se** $p = q$ **então**
    - **retorna** *verdadeiro*
  - **se** $inferida[p] = falso$ **então**
    - *inferida[p]* ← *verdadeiro*
    - **para cada** *cláusula c em que* $p \in PREMISSAS(c)$ **faça**
      - decremente $cont[c]$
      - **se** $cont[c] = 0$ **então**
        - Adicione a conclusão de *c* à *agenda*
- **retorna** *falso*;

BC:
1: $F \rightarrow E$
2: $P \wedge M \rightarrow F$
3: $D \wedge P \rightarrow M$
4: $A \wedge F \rightarrow P$
5: $A \wedge D \rightarrow P$
6: A
7: D

q: E p: E c: 4

cont:

1	2	3	4	5	6	7
0	0	0	0	0	0	0

inferida:

A	D	E	F	M	P
V	V	F	V	V	V

agenda: P

Lógica Proposicional e Inferência

Forward Chaining: Algoritmo

Função *FORWARD_CHAINING(BC, q)*: **boolean**

cont ← número de símbolos em cada premissa
inferida ← inicializada com *falso* para todo símbolo
agenda ← símbolos verdadeiros na *BC*

enquanto *agenda não estiver vazia* **faça**
  p ← retira o primeiro elemento de *agenda*
  se $p = q$ **então**
    retorna verdadeiro
  se $inferida[p] = falso$ **então**
    inferida[p] ← *verdadeiro*
    para cada *cláusula c em que*
    $p \in PREMISSAS(c)$ **faça**
      decremente $cont[c]$
      se $cont[c] = 0$ **então**
        Adicione a conclusão de *c* à *agenda*
retorna *falso*;

BC:
1: $F \rightarrow E$
2: $P \wedge M \rightarrow F$
3: $D \wedge P \rightarrow M$
4: $A \wedge F \rightarrow P$
5: $A \wedge D \rightarrow P$
6: A
7: D

q: E p: E c: 4

cont:

1	2	3	4	5	6	7
0	0	0	0	0	0	0

inferida:

A	D	E	F	M	P
V	V	F	V	V	V

agenda: P

Forward Chaining

- *Forward chaining* é consistente

Forward Chaining

- *Forward chaining* é consistente
  - Toda inferência é aplicação de Modus Ponens

Lógica Proposicional e Inferência

Forward Chaining

- *Forward chaining* é consistente
  - Toda inferência é aplicação de Modus Ponens
  - Assim, toda sentença atômica inferida é acarretada pela base

Forward Chaining

- *Forward chaining* é consistente
  - Toda inferência é aplicação de Modus Ponens
  - Assim, toda sentença atômica inferida é acarretada pela base
- Também é completo

Forward Chaining

- *Forward chaining* é consistente
  - Toda inferência é aplicação de Modus Ponens
  - Assim, toda sentença atômica inferida é acarretada pela base
- Também é completo
  - Toda sentença atômica acarretada pela base será derivada

Lógica Proposicional e Inferência

Forward Chaining

- *Forward chaining* é consistente
  - Toda inferência é aplicação de Modus Ponens
  - Assim, toda sentença atômica inferida é acarretada pela base
- Também é completo
  - Toda sentença atômica acarretada pela base será derivada
- É um exemplo de **raciocínio guiado por dados**

Forward Chaining

- *Forward chaining* é consistente
  - Toda inferência é aplicação de Modus Ponens
  - Assim, toda sentença atômica inferida é acarretada pela base
- Também é completo
  - Toda sentença atômica acarretada pela base será derivada
- É um exemplo de **raciocínio guiado por dados**
  - Raciocínio em que o foco da atenção começa com os dados conhecidos

Backward Chaining

Backward Chaining

- Também se baseia em Modus Ponens

Backward Chaining

- Também se baseia em Modus Ponens
  - Porém caminha em direção contrária ao *forward chaining*, voltando nas regras a partir da query Q

Lógica Proposicional e Inferência

Backward Chaining

- Também se baseia em Modus Ponens
  - Porém caminha em direção contrária ao *forward chaining*, voltando nas regras a partir da query Q
- Para provar Q por *Backward Chaining*

Backward Chaining

- Também se baseia em Modus Ponens
  - Porém caminha em direção contrária ao *forward chaining*, voltando nas regras a partir da query Q
- Para provar Q por *Backward Chaining*
  - Comece do objetivo Q

Backward Chaining

- Também se baseia em Modus Ponens
  - Porém caminha em direção contrária ao *forward chaining*, voltando nas regras a partir da query Q
- Para provar Q por *Backward Chaining*
  - Comece do objetivo Q
  - Se Q for verdadeira, então não precisa fazer mais nada

Lógica Proposicional e Inferência

Backward Chaining

- Também se baseia em Modus Ponens
  - Porém caminha em direção contrária ao *forward chaining*, voltando nas regras a partir da query Q
- Para provar Q por *Backward Chaining*
  - Comece do objetivo Q
  - Se Q for verdadeira, então não precisa fazer mais nada
  - Senão, encontre as implicações na BC que concluem Q

Lógica Proposicional e Inferência

Backward Chaining

- Também se baseia em Modus Ponens
  - Porém caminha em direção contrária ao *forward chaining*, voltando nas regras a partir da query Q
- Para provar Q por *Backward Chaining*
  - Comece do objetivo Q
  - Se Q for verdadeira, então não precisa fazer mais nada
  - Senão, encontre as implicações na BC que concluem Q
  - Se todas as premissas de uma dessas implicações puderem ser provadas verdadeiras (por *backward chaining*), então Q é verdadeira

Lógica Proposicional e Inferência

Forward Chaining

$$F \rightarrow E$$
$$P \wedge M \rightarrow F$$
$$D \wedge P \rightarrow M$$
$$A \wedge F \rightarrow P$$
$$A \wedge D \rightarrow P$$
$$A$$
$$D$$

Backward Chaining

Lógica Proposicional e Inferência

Forward Chaining

Começamos com fatos conhecidos

$F \rightarrow E$
$P \wedge M \rightarrow F$
$D \wedge P \rightarrow M$
$A \wedge F \rightarrow P$
$A \wedge D \rightarrow P$
A
D

Backward Chaining

Lógica Proposicional e Inferência

Forward Chaining

De A e D derivamos P

$$\frac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{F \rightarrow E}{P \wedge M \rightarrow F}}{D \wedge P \rightarrow M}}{A \wedge F \rightarrow P}}{\color{red}{A \wedge D \rightarrow P}}}{A}}{D}}{P}$$

Backward Chaining

Lógica Proposicional e Inferência

Forward Chaining

De P e D derivamos M

$F \to E$
$P \wedge M \to F$
$D \wedge P \to M$
$A \wedge F \to P$
$A \wedge D \to P$
A
D
P
M

Backward Chaining

Lógica Proposicional e Inferência

Forward Chaining

De P e M derivamos F

$F \rightarrow E$
$P \wedge M \rightarrow F$
$D \wedge P \rightarrow M$
$A \wedge F \rightarrow P$
$A \wedge D \rightarrow P$
A
D
P
M
F

Backward Chaining

Lógica Proposicional e Inferência

Forward Chaining

De F temos E

$F \rightarrow E$
$P \wedge M \rightarrow F$
$D \wedge P \rightarrow M$
$A \wedge F \rightarrow P$
$A \wedge D \rightarrow P$
A
D
P
M
F

Backward Chaining

Lógica Proposicional e Inferência

Forward Chaining

De F temos E

$F \rightarrow E$
$P \wedge M \rightarrow F$
$D \wedge P \rightarrow M$
$A \wedge F \rightarrow P$
$A \wedge D \rightarrow P$
A
D
P
M
F

Backward Chaining

Buscamos regra com $\rightarrow E$.

Precisamos de F

$F \rightarrow E$
$P \wedge M \rightarrow F$
$D \wedge P \rightarrow M$
$A \wedge F \rightarrow P$
$A \wedge D \rightarrow P$
A
D

Lógica Proposicional e Inferência

Forward Chaining

De F temos E

$F \to E$
$P \land M \to F$
$D \land P \to M$
$A \land F \to P$
$A \land D \to P$
A
D
P
M
F

Backward Chaining

Buscamos regra com $\to F$.
Faltam P e M

$F \to E$
$P \land M \to F$
$D \land P \to M$
$A \land F \to P$
$A \land D \to P$
A
D

Lógica Proposicional e Inferência

Forward Chaining

De F temos E

$F \rightarrow E$
$P \wedge M \rightarrow F$
$D \wedge P \rightarrow M$
$A \wedge F \rightarrow P$
$A \wedge D \rightarrow P$
A
D
P
M
F

Backward Chaining

Buscamos regra com $\rightarrow P$.
Faltam A e F

$F \rightarrow E$
$P \wedge M \rightarrow F$
$D \wedge P \rightarrow M$
$A \wedge F \rightarrow P$
$A \wedge D \rightarrow P$
A
D

Lógica Proposicional e Inferência

Forward Chaining

De F temos E

$F \rightarrow E$
$P \wedge M \rightarrow F$
$D \wedge P \rightarrow M$
$A \wedge F \rightarrow P$
$A \wedge D \rightarrow P$
A
D
P
M
F

Backward Chaining

Conhecemos A, falta F

$F \rightarrow E$
$P \wedge M \rightarrow F$
$D \wedge P \rightarrow M$
$A \wedge F \rightarrow P$
$A \wedge D \rightarrow P$
A
D

Lógica Proposicional e Inferência

Forward Chaining

De F temos E

$$\begin{array}{c} F \rightarrow E \\ \hline P \wedge M \rightarrow F \\ \hline D \wedge P \rightarrow M \\ \hline A \wedge F \rightarrow P \\ \hline A \wedge D \rightarrow P \\ \hline A \\ \hline D \\ \hline P \\ \hline M \\ \hline F \end{array}$$

Backward Chaining

Cuidado! Se seguirmos com F teremos um laço!

$$\begin{array}{c} \color{blue}{F} \rightarrow E \\ \hline \color{blue}{P} \color{red}{\wedge} \color{blue}{M} \color{red}{\rightarrow F} \\ \hline D \wedge P \rightarrow M \\ \hline \color{red}{A \wedge} \color{blue}{F} \color{red}{\rightarrow P} \\ \hline A \wedge D \rightarrow P \\ \hline A \\ \hline D \end{array}$$

Lógica Proposicional e Inferência

Forward Chaining

De F temos E

$F \rightarrow E$
$P \wedge M \rightarrow F$
$D \wedge P \rightarrow M$
$A \wedge F \rightarrow P$
$A \wedge D \rightarrow P$
A
D
P
M
F

Backward Chaining

Tentamos a próxima regra com $\rightarrow P$. Falta D

$F \rightarrow E$
$P \wedge M \rightarrow F$
$D \wedge P \rightarrow M$
$A \wedge F \rightarrow P$
$A \wedge D \rightarrow P$
A
D

Lógica Proposicional e Inferência

Forward Chaining

De F temos E

$F \to E$
$P \wedge M \to F$
$D \wedge P \to M$
$A \wedge F \to P$
$A \wedge D \to P$
A
D
P
M
F

Backward Chaining

Conhecemos D, então conhecemos P. Falta F

$F \to E$
$P \wedge M \to F$
$D \wedge P \to M$
$A \wedge F \to P$
$A \wedge D \to P$
A
D
P

Lógica Proposicional e Inferência

Forward Chaining

De F temos E

$F \to E$
$P \wedge M \to F$
$D \wedge P \to M$
$A \wedge F \to P$
$A \wedge D \to P$
A
D
P
M
F

Backward Chaining

Voltamos à regra $\to F$. Falta M

$F \to E$
$P \wedge M \to F$
$D \wedge P \to M$
$A \wedge F \to P$
$A \wedge D \to P$
A
D
P

Lógica Proposicional e Inferência

Forward Chaining

De F temos E

$F \to E$
$P \wedge M \to F$
$D \wedge P \to M$
$A \wedge F \to P$
$A \wedge D \to P$
A
D
P
M
F

Backward Chaining

Buscamos regra com $\to M$. Faltam D e P

$\color{blue}{F} \to E$
$P \wedge \color{blue}{M} \to F$
$\color{red}{D \wedge P \to M}$
$A \wedge \color{blue}{F} \to P$
$A \wedge D \to P$
A
D
P

Lógica Proposicional e Inferência

Forward Chaining

De F temos E

$F \to E$
$P \wedge M \to F$
$D \wedge P \to M$
$A \wedge F \to P$
$A \wedge D \to P$
A
D
P
M
F

Backward Chaining

Conhecemos D e P, então temos M. Falta F

$F \to E$
$P \wedge M \to F$
$D \wedge P \to M$
$A \wedge F \to P$
$A \wedge D \to P$
A
D
P
M

Lógica Proposicional e Inferência

Forward Chaining

De F temos E

$F \rightarrow E$
$P \wedge M \rightarrow F$
$D \wedge P \rightarrow M$
$A \wedge F \rightarrow P$
$A \wedge D \rightarrow P$
A
D
P
M
F

Backward Chaining

Voltamos à regra $\rightarrow F$. Temos P e M. Temos F

$F \rightarrow E$
$P \wedge M \rightarrow F$
$D \wedge P \rightarrow M$
$A \wedge F \rightarrow P$
$A \wedge D \rightarrow P$
A
D
P
M
F

Lógica Proposicional e Inferência

Forward Chaining

De F temos E

$F \rightarrow E$
$P \wedge M \rightarrow F$
$D \wedge P \rightarrow M$
$A \wedge F \rightarrow P$
$A \wedge D \rightarrow P$
A
D
P
M
F

Backward Chaining

Se temos F, temos E

$F \rightarrow E$
$P \wedge M \rightarrow F$
$D \wedge P \rightarrow M$
$A \wedge F \rightarrow P$
$A \wedge D \rightarrow P$
A
D
P
M
F

Backward Chaining

- Forma de **raciocínio dirigido por objetivo**

Backward Chaining

- Forma de **raciocínio dirigido por objetivo**
- Custo menor que o tamanho da base de dados:

Backward Chaining

- Forma de **raciocínio dirigido por objetivo**
- Custo menor que o tamanho da base de dados:
  - Aborda somente fatos relevantes

Lógica Proposicional e Inferência

Backward Chaining

- Forma de **raciocínio dirigido por objetivo**
- Custo menor que o tamanho da base de dados:
  - Aborda somente fatos relevantes
  - *Forward Chaining* pode gastar muito tempo derivando informação irrelevante para o problema

Backward Chaining

- Forma de **raciocínio dirigido por objetivo**
- Custo menor que o tamanho da base de dados:
  - Aborda somente fatos relevantes
  - *Forward Chaining* pode gastar muito tempo derivando informação irrelevante para o problema
- Por outro lado, há que se cuidar para evitar laços

Backward Chaining

- Forma de **raciocínio dirigido por objetivo**
- Custo menor que o tamanho da base de dados:
  - Aborda somente fatos relevantes
  - *Forward Chaining* pode gastar muito tempo derivando informação irrelevante para o problema
- Por outro lado, há que se cuidar para evitar laços
- Além de verificar constantemente se as premissas já não foram determinadas

Backward Chaining

- Forma de **raciocínio dirigido por objetivo**
- Custo menor que o tamanho da base de dados:
  - Aborda somente fatos relevantes
  - *Forward Chaining* pode gastar muito tempo derivando informação irrelevante para o problema
- Por outro lado, há que se cuidar para evitar laços
- Além de verificar constantemente se as premissas já não foram determinadas
  - Antes de verificar novas regras que as derivem

Lógica Proposicional e Inferência

Resolução

- *Backward* e *Forward Chaining* dependem da base de conhecimento conter apenas cláusulas definidas

Lógica Proposicional e Inferência

Resolução

- *Backward* e *Forward Chaining* dependem da base de conhecimento conter apenas cláusulas definidas
- E se este não for o caso?

Lógica Proposicional e Inferência

Resolução

- *Backward* e *Forward Chaining* dependem da base de conhecimento conter apenas cláusulas definidas
- E se este não for o caso?
  - Toda sentença em lógica proposicional é logicamente equivalente a uma conjunção de disjunções de literais

Lógica Proposicional e Inferência

Resolução

- *Backward* e *Forward Chaining* dependem da base de conhecimento conter apenas cláusulas definidas
- E se este não for o caso?
  - Toda sentença em lógica proposicional é logicamente equivalente a uma conjunção de disjunções de literais
    - Ou seja, a uma conjunção de cláusulas

Lógica Proposicional e Inferência

Resolução

- *Backward* e *Forward Chaining* dependem da base de conhecimento conter apenas cláusulas definidas
- E se este não for o caso?
  - Toda sentença em lógica proposicional é logicamente equivalente a uma conjunção de disjunções de literais
    - Ou seja, a uma conjunção de cláusulas
  - Exemplo:

Resolução

- *Backward* e *Forward Chaining* dependem da base de conhecimento conter apenas cláusulas definidas
- E se este não for o caso?
  - Toda sentença em lógica proposicional é logicamente equivalente a uma conjunção de disjunções de literais
    - Ou seja, a uma conjunção de cláusulas
  - Exemplo:
    - $(A \vee B \vee \neg C) \wedge (B \vee D) \wedge (\neg A) \wedge (B \vee C)$

Lógica Proposicional e Inferência

Resolução

- *Backward* e *Forward Chaining* dependem da base de conhecimento conter apenas cláusulas definidas
- E se este não for o caso?
  - Toda sentença em lógica proposicional é logicamente equivalente a uma conjunção de disjunções de literais
    - Ou seja, a uma conjunção de cláusulas
  - Exemplo:
    - $(A \lor B \lor \neg C) \land (B \lor D) \land (\neg A) \land (B \lor C)$
    - $(A \lor B \lor \neg C)$

Resolução

- *Backward* e *Forward Chaining* dependem da base de conhecimento conter apenas cláusulas definidas
- E se este não for o caso?
  - Toda sentença em lógica proposicional é logicamente equivalente a uma conjunção de disjunções de literais
    - Ou seja, a uma conjunção de cláusulas
  - Exemplo:
    - $(A \vee B \vee \neg C) \wedge (B \vee D) \wedge (\neg A) \wedge (B \vee C)$
    - $(A \vee B \vee \neg C)$
  - **Forma Normal Conjuntiva** – FNC

Resolução: Conversão para FNC

Resolução: Conversão para FNC

1. Elimine implicações e equivalências usando suas definições:

Resolução: Conversão para FNC

1. Elimine implicações e equivalências usando suas definições:
   - $p \rightarrow q \equiv \neg p \vee q$

Resolução: Conversão para FNC

1. Elimine implicações e equivalências usando suas definições:
   - $p \rightarrow q \equiv \neg p \vee q$
   - $p \leftrightarrow q \equiv (p \rightarrow q) \wedge (q \rightarrow p)$

Resolução: Conversão para FNC

1. Elimine implicações e equivalências usando suas definições:
   - $p \rightarrow q \equiv \neg p \vee q$
   - $p \leftrightarrow q \equiv (p \rightarrow q) \wedge (q \rightarrow p)$
2. Coloque as negações “para dentro dos parênteses”

Resolução: Conversão para FNC

1. Elimine implicações e equivalências usando suas definições:
    - $p \rightarrow q \equiv \neg p \vee q$
    - $p \leftrightarrow q \equiv (p \rightarrow q) \wedge (q \rightarrow p)$
2. Coloque as negações "para dentro dos parênteses"
    - $\neg(\neg p) \equiv p$

Resolução: Conversão para FNC

1. Elimine implicações e equivalências usando suas definições:
   - $p \rightarrow q \equiv \neg p \vee q$
   - $p \leftrightarrow q \equiv (p \rightarrow q) \wedge (q \rightarrow p)$
2. Coloque as negações “para dentro dos parênteses”
   - $\neg(\neg p) \equiv p$
   - $\neg(p \vee q) \equiv (\neg p \wedge \neg q)$ (De Morgan)

Resolução: Conversão para FNC

1. Elimine implicações e equivalências usando suas definições:
   - $p \rightarrow q \equiv \neg p \vee q$
   - $p \leftrightarrow q \equiv (p \rightarrow q) \wedge (q \rightarrow p)$
2. Coloque as negações "para dentro dos parênteses"
   - $\neg(\neg p) \equiv p$
   - $\neg(p \vee q) \equiv (\neg p \wedge \neg q)$ (De Morgan)
   - $\neg(p \wedge q) \equiv (\neg p \vee \neg q)$ (De Morgan)

Resolução: Conversão para FNC

1. Elimine implicações e equivalências usando suas definições:
   - $p \to q \equiv \neg p \vee q$
   - $p \leftrightarrow q \equiv (p \to q) \wedge (q \to p)$
2. Coloque as negações "para dentro dos parênteses"
   - $\neg(\neg p) \equiv p$
   - $\neg(p \vee q) \equiv (\neg p \wedge \neg q)$ (De Morgan)
   - $\neg(p \wedge q) \equiv (\neg p \vee \neg q)$ (De Morgan)
3. Aplique a propriedade distributiva ($\vee$ sobre $\wedge$):

Lógica Proposicional e Inferência

Resolução: Conversão para FNC

1. Elimine implicações e equivalências usando suas definições:
   - $p \rightarrow q \equiv \neg p \vee q$
   - $p \leftrightarrow q \equiv (p \rightarrow q) \wedge (q \rightarrow p)$
2. Coloque as negações "para dentro dos parênteses"
   - $\neg(\neg p) \equiv p$
   - $\neg(p \vee q) \equiv (\neg p \wedge \neg q)$ (De Morgan)
   - $\neg(p \wedge q) \equiv (\neg p \vee \neg q)$ (De Morgan)
3. Aplique a propriedade distributiva ($\vee$ sobre $\wedge$):
   - $p \vee (q \wedge r) \equiv (p \vee q) \wedge (p \vee r)$

Conversão para FNC: Exemplo

- $A \leftrightarrow (B \vee C)$

Conversão para FNC: Exemplo

- $A \leftrightarrow (B \vee C)$
- Elimine $\leftrightarrow$:

Conversão para FNC: Exemplo

- $A \leftrightarrow (B \vee C)$
- Elimine $\leftrightarrow$:
  - $(A \rightarrow (B \vee C)) \wedge ((B \vee C) \rightarrow A)$

Conversão para FNC: Exemplo

- $A \leftrightarrow (B \lor C)$
- Elimine $\leftrightarrow$:
  - $(A \rightarrow (B \lor C)) \land ((B \lor C) \rightarrow A)$
- Elimine $\rightarrow$:

Conversão para FNC: Exemplo

- $A \leftrightarrow (B \vee C)$
- Elimine $\leftrightarrow$:
  - $(A \rightarrow (B \vee C)) \wedge ((B \vee C) \rightarrow A)$
- Elimine $\rightarrow$:
  - $(\neg A \vee (B \vee C)) \wedge (\neg(B \vee C) \vee A)$

Lógica Proposicional e Inferência

Conversão para FNC: Exemplo

- $A \leftrightarrow (B \vee C)$
- Elimine $\leftrightarrow$:
  - $(A \rightarrow (B \vee C)) \wedge ((B \vee C) \rightarrow A)$
- Elimine $\rightarrow$:
  - $(\neg A \vee (B \vee C)) \wedge (\neg(B \vee C) \vee A)$
- Mova o $\neg$:

Conversão para FNC: Exemplo

- $A \leftrightarrow (B \lor C)$
- Elimine $\leftrightarrow$:
  - $(A \rightarrow (B \lor C)) \land ((B \lor C) \rightarrow A)$
- Elimine $\rightarrow$:
  - $(\neg A \lor (B \lor C)) \land (\neg(B \lor C) \lor A)$
- Mova o $\neg$:
  - $(\neg A \lor B \lor C) \land ((\neg B \land \neg C) \lor A)$

Conversão para FNC: Exemplo

- $A \leftrightarrow (B \vee C)$
- Elimine $\leftrightarrow$:
  - $(A \rightarrow (B \vee C)) \wedge ((B \vee C) \rightarrow A)$
- Elimine $\rightarrow$:
  - $(\neg A \vee (B \vee C)) \wedge (\neg(B \vee C) \vee A)$
- Mova o $\neg$:
  - $(\neg A \vee B \vee C) \wedge ((\neg B \wedge \neg C) \vee A)$
- Distribua $\vee$ sobre $\wedge$:

Conversão para FNC: Exemplo

- $A \leftrightarrow (B \vee C)$
- Elimine $\leftrightarrow$:
  - $(A \rightarrow (B \vee C)) \wedge ((B \vee C) \rightarrow A)$
- Elimine $\rightarrow$:
  - $(\neg A \vee (B \vee C)) \wedge (\neg(B \vee C) \vee A)$
- Mova o $\neg$:
  - $(\neg A \vee B \vee C) \wedge ((\neg B \wedge \neg C) \vee A)$
- Distribua $\vee$ sobre $\wedge$:
  - $(\neg A \vee B \vee C) \wedge (A \vee \neg B) \wedge (A \vee \neg C)$

Resolução: Provas na FNC

Resolução: Provas na FNC

- Prova por absurdo (ou contradição)

Resolução: Provas na FNC

- Prova por absurdo (ou contradição)
  - Negue a conclusão à qual se deseja chegar

Resolução: Provas na FNC

- Prova por absurdo (ou contradição)
  - Negue a conclusão à qual se deseja chegar
  - Demonstre que isso implica violar alguma proposição estabelecida – uma contradição

Resolução: Provas na FNC

- Prova por absurdo (ou contradição)
  - Negue a conclusão à qual se deseja chegar
  - Demonstre que isso implica violar alguma proposição estabelecida – uma contradição
  - Como?

Resolução: Provas na FNC

- Prova por absurdo (ou contradição)
  - Negue a conclusão à qual se deseja chegar
  - Demonstre que isso implica violar alguma proposição estabelecida – uma contradição
  - Como?
- **Regra da resolução**:

Resolução: Provas na FNC

- Prova por absurdo (ou contradição)
  - Negue a conclusão à qual se deseja chegar
  - Demonstre que isso implica violar alguma proposição estabelecida – uma contradição
  - Como?
- **Regra da resolução**:
  - $$\frac{\begin{array}{c} A \vee B \\ \neg B \end{array}}{A} \qquad\qquad \frac{\begin{array}{c} A \vee B \\ \neg B \vee C \end{array}}{A \vee C}$$

Lógica Proposicional e Inferência

Resolução: Provas na FNC

- Prova por absurdo (ou contradição)
  - Negue a conclusão à qual se deseja chegar
  - Demonstre que isso implica violar alguma proposição estabelecida – uma contradição
  - Como?
- **Regra da resolução**:
  - $$\frac{\begin{array}{c} A \vee B \\ \neg B \end{array}}{A} \qquad\qquad \frac{\begin{array}{c} A \vee B \\ \neg B \vee C \end{array}}{A \vee C}$$

Dizemos que o literal B na primeira resolve com o literal $\neg B$ na segunda, para dar o **resolvente** $A \vee C$

Regra da Resolução

- Forma geral:

$$\frac{\begin{array}{c} l_1 \vee \ldots \vee l_k \\ m_1 \vee \ldots \vee m_n \end{array}}{l_1 \vee \ldots \vee l_{i-1} \vee l_{i+1} \vee \ldots \vee l_k \vee m_1 \vee \ldots \vee m_{j-1} \vee m_{j+1} \vee \ldots \vee m_n}$$

Onde l_i e m_j são literais complementares.

Regra da Resolução

- Forma geral:

$$\frac{\begin{array}{c} l_1 \vee \ldots \vee l_k \\ m_1 \vee \ldots \vee m_n \end{array}}{l_1 \vee \ldots \vee l_{i-1} \vee l_{i+1} \vee \ldots \vee l_k \vee m_1 \vee \ldots \vee m_{j-1} \vee m_{j+1} \vee \ldots \vee m_n}$$

Onde l_i e m_j são literais complementares.

- Literais Complementares:

Regra da Resolução

- Forma geral:

$$\frac{\begin{array}{c} l_1 \vee \ldots \vee l_k \\ m_1 \vee \ldots \vee m_n \end{array}}{l_1 \vee \ldots \vee l_{i-1} \vee l_{i+1} \vee \ldots \vee l_k \vee m_1 \vee \ldots \vee m_{j-1} \vee m_{j+1} \vee \ldots \vee m_n}$$

  Onde l_i e m_j são literais complementares.

- Literais Complementares:
  - Dois literais são complementares quando um for a negação do outro

Lógica Proposicional e Inferência

Regra da Resolução

- Forma geral:

$$\frac{\begin{array}{c} l_1 \vee \ldots \vee l_k \\ m_1 \vee \ldots \vee m_n \end{array}}{l_1 \vee \ldots \vee l_{i-1} \vee l_{i+1} \vee \ldots \vee l_k \vee m_1 \vee \ldots \vee m_{j-1} \vee m_{j+1} \vee \ldots \vee m_n}$$

  Onde l_i e m_j são literais complementares.

- Literais Complementares:
  - Dois literais são complementares quando um for a negação do outro

- A aplicação de uma única regra (resolução) pode derivar qualquer conclusão em lógica proposicional

Prova por Resolução: Metodologia

Prova por Resolução: Metodologia

- Converta todas as sentenças para FNC

Prova por Resolução: Metodologia

- Converta todas as sentenças para FNC
- Negue a conclusão desejada (em FNC)

Lógica Proposicional e Inferência

Prova por Resolução: Metodologia

- Converta todas as sentenças para FNC
- Negue a conclusão desejada (em FNC)
- Aplique a Regra da Resolução a cada par de cláusulas com literais complementares, adicionando as cláusulas produzidas à base, até que:

Prova por Resolução: Metodologia

- Converta todas as sentenças para FNC
- Negue a conclusão desejada (em FNC)
- Aplique a Regra da Resolução a cada par de cláusulas com literais complementares, adicionando as cláusulas produzidas à base, até que:
  - Se obtenha o resultado Falso (contradição), caso em que a conclusão segue dos axiomas; ou

Prova por Resolução: Metodologia

- Converta todas as sentenças para FNC
- Negue a conclusão desejada (em FNC)
- Aplique a Regra da Resolução a cada par de cláusulas com literais complementares, adicionando as cláusulas produzidas à base, até que:
  - Se obtenha o resultado Falso (contradição), caso em que a conclusão segue dos axiomas; ou
  - Não se possa mais aplicá-la, caso em que a conclusão não pode ser provada a partir dos axiomas

Exemplo: Prove R

Base de Conhecimento:

1	$(P \rightarrow Q) \rightarrow Q$
2	$(P \rightarrow P) \rightarrow R$
3	$(R \rightarrow S) \rightarrow \neg(S \rightarrow Q)$

Exemplo: Prove R

Converta cada oração para FNC, simplificando-a:

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

Base de Conhecimento:

1	$(P \rightarrow Q) \rightarrow Q$
2	$(P \rightarrow P) \rightarrow R$
3	$(R \rightarrow S) \rightarrow \neg(S \rightarrow Q)$

Exemplo: Prove R

Converta cada oração para FNC, simplificando-a:

Base de Conhecimento:

1	$(P \to Q) \to Q$
2	$(P \to P) \to R$
3	$(R \to S) \to \neg(S \to Q)$

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

$$\neg(\neg P \vee P) \vee R$$
$$(P \wedge \neg P) \vee R$$
$$(P \vee R) \wedge (\neg P \vee R)$$

Exemplo: Prove R

Converta cada oração para FNC, simplificando-a:

Base de Conhecimento:

1	$(P \rightarrow Q) \rightarrow Q$
2	$(P \rightarrow P) \rightarrow R$
3	$(R \rightarrow S) \rightarrow \neg(S \rightarrow Q)$

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

$$\neg(\neg P \vee P) \vee R$$
$$(P \wedge \neg P) \vee R$$
$$(P \vee R) \wedge (\neg P \vee R)$$

$$\neg(\neg R \vee S) \vee \neg(\neg S \vee Q)$$
$$(R \wedge \neg S) \vee (S \wedge \neg Q)$$
$$(R \vee S) \wedge (\neg S \vee S) \wedge (R \vee \neg Q) \wedge (\neg S \vee \neg Q)$$
$$(R \vee S) \wedge (R \vee \neg Q) \wedge (\neg S \vee \neg Q)$$

Exemplo: Prove R

Copie cada sentença para uma base auxiliar, incluindo a negação do objetivo:

1	$(P \rightarrow Q) \rightarrow Q$
2	$(P \rightarrow P) \rightarrow R$
3	$(R \rightarrow S) \rightarrow \neg(S \rightarrow Q)$

Lógica Proposicional e Inferência

Exemplo: Prove R

Copie cada sentença para uma base auxiliar, incluindo a negação do objetivo:

1	$(P \rightarrow Q) \rightarrow Q$
2	$(P \rightarrow P) \rightarrow R$
3	$(R \rightarrow S) \rightarrow \neg(S \rightarrow Q)$

1	$P \vee Q$	
2	$P \vee R$	
3	$\neg P \vee R$	
4	$R \vee S$	
5	$R \vee \neg Q$	
6	$\neg S \vee \neg Q$	
7	$\neg R$	Neg
8		
9		
10		
11		
12		

Lógica Proposicional e Inferência

Exemplo: Prove R

Copie cada sentença para uma base auxiliar, incluindo a negação do objetivo:

1	$(P \rightarrow Q) \rightarrow Q$
2	$(P \rightarrow P) \rightarrow R$
3	$(R \rightarrow S) \rightarrow \neg(S \rightarrow Q)$

1	$P \vee Q$	
2	$P \vee R$	
3	$\neg P \vee R$	
4	$R \vee S$	
5	$R \vee \neg Q$	
6	$\neg S \vee \neg Q$	
7	$\neg R$	Neg
8		
9		
10		
11		
12		

Cada cláusula unida a outra por um E foi posta em uma linha separada

Exemplo: Prove R

Aplicando apenas a Regra da Resolução, uma possível prova seria:

1	$P \vee Q$	
2	$P \vee R$	
3	$\neg P \vee R$	
4	$R \vee S$	
5	$R \vee \neg Q$	
6	$\neg S \vee \neg Q$	
7	$\neg R$	Neg
8		
9		
10		
11		
12		

Exemplo: Prove R

Aplicando apenas a Regra da Resolução, uma possível prova seria:

1	$P \vee Q$	
2	$P \vee R$	
3	$\neg P \vee R$	
4	$R \vee S$	
5	$R \vee \neg Q$	
6	$\neg S \vee \neg Q$	
7	$\neg R$	Neg
8	S	4,7
9		
10		
11		
12		

Exemplo: Prove R

Aplicando apenas a Regra da Resolução, uma possível prova seria:

1	$P \vee Q$	
2	$P \vee R$	
3	$\neg P \vee R$	
4	$R \vee S$	
5	$R \vee \neg Q$	
6	$\neg S \vee \neg Q$	
7	$\neg R$	Neg
8	S	4,7
9	$\neg Q$	6,8
10		
11		
12		

Exemplo: Prove R

Aplicando apenas a Regra da Resolução, uma possível prova seria:

1	$P \vee Q$	
2	$P \vee R$	
3	$\neg P \vee R$	
4	$R \vee S$	
5	$R \vee \neg Q$	
6	$\neg S \vee \neg Q$	
7	$\neg R$	Neg
8	S	4,7
9	$\neg Q$	6,8
10	P	1,9
11		
12		

Exemplo: Prove R

Aplicando apenas a Regra da Resolução, uma possível prova seria:

1	$P \vee Q$	
2	$P \vee R$	
3	$\neg P \vee R$	
4	$R \vee S$	
5	$R \vee \neg Q$	
6	$\neg S \vee \neg Q$	
7	$\neg R$	Neg
8	S	4,7
9	$\neg Q$	6,8
10	P	1,9
11	R	3,10
12		

Exemplo: Prove R

Aplicando apenas a Regra da Resolução, uma possível prova seria:

1	$P \vee Q$	
2	$P \vee R$	
3	$\neg P \vee R$	
4	$R \vee S$	
5	$R \vee \neg Q$	
6	$\neg S \vee \neg Q$	
7	$\neg R$	Neg
8	S	4,7
9	$\neg Q$	6,8
10	P	1,9
11	R	3,10
12	*falso*	7,11

Exemplo: Prove R

Aplicando apenas a Regra da Resolução, uma possível prova seria:

A prova acabou chegando a uma contradição: Se aplicarmos a regra a 7 e 11 teremos $R \land \neg R$ (a contradição), o que nos leva a incluir Falso na base (nosso critério de parada)

1	$P \lor Q$	
2	$P \lor R$	
3	$\neg P \lor R$	
4	$R \lor S$	
5	$R \lor \neg Q$	
6	$\neg S \lor \neg Q$	
7	$\neg R$	Neg
8	S	4,7
9	$\neg Q$	6,8
10	P	1,9
11	R	3,10
12	*falso*	7,11

Cuidado!

- Base:

	Cláusula	*Derivação*
1	P	–
2	$\neg P$	–
3		
4		

Prove Z

Lógica Proposicional e Inferência

Cuidado!

- Base:

	Cláusula	*Derivação*
1	P	–
2	$\neg P$	–
3	$\neg Z$	Neg
4		

Prove Z

Cuidado!

- Base:

	Cláusula	*Derivação*
1	P	–
2	$\neg P$	–
3	$\neg Z$	Neg
4	*falso*	1,2

Prove Z

Lógica Proposicional e Inferência

Cuidado!

- Base:

	Cláusula	*Derivação*
1	P	–
2	$\neg P$	–
3	$\neg Z$	Neg
4	*falso*	1,2

Prove Z

- Qualquer conclusão segue de uma contradição

Cuidado!

- Base:

	Cláusula	*Derivação*
1	P	–
2	$\neg P$	–
3	$\neg Z$	Neg
4	*falso*	1,2

Prove Z

- Qualquer conclusão segue de uma contradição
  - Faz sentido, uma vez que $P \wedge \neg P \rightarrow Z$ é verdadeiro

Cuidado!

- Base:

	Cláusula	*Derivação*
1	P	–
2	$\neg P$	–
3	$\neg Z$	Neg
4	*falso*	1,2

Prove Z

- Qualquer conclusão segue de uma contradição
  - Faz sentido, uma vez que $P \wedge \neg P \rightarrow Z$ é verdadeiro
  - Isso torna sistemas lógicos frágeis

Cuidado!

- Base:

	Cláusula	*Derivação*
1	P	–
2	$\neg P$	–
3	$\neg Z$	Neg
4	*falso*	1,2

Prove Z

- Qualquer conclusão segue de uma contradição
  - Faz sentido, uma vez que $P \wedge \neg P \rightarrow Z$ é verdadeiro
  - Isso torna sistemas lógicos frágeis
  - E é por isso que a base não pode conter tais contradições!!!

Em suma...

- Vantagens:

Em suma...

- Vantagens:
  - É **declarativa**: elementos sintáticos correspondem a fatos do mundo

Lógica Proposicional

Em suma...

- Vantagens:
  - É **declarativa**: elementos sintáticos correspondem a fatos do mundo
  - Lida com informação parcial: via disjunção ($\vee$) e negação

Lógica Proposicional

Em suma...

- Vantagens:
  - É **declarativa**: elementos sintáticos correspondem a fatos do mundo
  - Lida com informação parcial: via disjunção ($\vee$) e negação
  - É **composicional**: o significado de uma sentença é uma função do significado de suas partes

Lógica Proposicional

Em suma...

- Vantagens:
  - É **declarativa**: elementos sintáticos correspondem a fatos do mundo
  - Lida com informação parcial: via disjunção ($\vee$) e negação
  - É **composicional**: o significado de uma sentença é uma função do significado de suas partes
  - É **independente do contexto**:

Lógica Proposicional

Em suma...

- Vantagens:
  - É **declarativa**: elementos sintáticos correspondem a fatos do mundo
  - Lida com informação parcial: via disjunção ($\vee$) e negação
  - É **composicional**: o significado de uma sentença é uma função do significado de suas partes
  - É **independente do contexto**:
    - Verdadeiro é verdadeiro independentemente do contexto

Lógica Proposicional

Em suma...

- Vantagens:
  - É **declarativa**: elementos sintáticos correspondem a fatos do mundo
  - Lida com informação parcial: via disjunção ($\vee$) e negação
  - É **composicional**: o significado de uma sentença é uma função do significado de suas partes
  - É **independente do contexto**:
    - Verdadeiro é verdadeiro independentemente do contexto
    - Diferentemente de uma língua natural, onde esse valor depende do contexto

Lógica Proposicional

Em suma...

- Desvantagens:

Lógica Proposicional

Em suma...

- Desvantagens:
  - Como representar conceitos, quantidades, coisas e pessoas?

Em suma...

- Desvantagens:
  - Como representar conceitos, quantidades, coisas e pessoas?
  - Como representar "casa vizinha à minha"?

Em suma...

- Desvantagens:
  - Como representar conceitos, quantidades, coisas e pessoas?
  - Como representar "casa vizinha à minha"?
    - Criando uma regra para cada casa vizinha

Lógica Proposicional

Em suma...

- Desvantagens:
  - Como representar conceitos, quantidades, coisas e pessoas?
  - Como representar "casa vizinha à minha"?
    - Criando uma regra para cada casa vizinha
    - Não há como descrever concisamente um ambiente com muitos objetos

Lógica Proposicional

Em suma...

- Desvantagens:
  - Como representar conceitos, quantidades, coisas e pessoas?
  - Como representar "casa vizinha à minha"?
    - Criando uma regra para cada casa vizinha
    - Não há como descrever concisamente um ambiente com muitos objetos
  - Como representar "todo homem é mortal", sem enumerar cada homem e ligá-lo à mortalidade?

Lógica Proposicional

Em suma...

- Desvantagens:
  - Como representar conceitos, quantidades, coisas e pessoas?
  - Como representar "casa vizinha à minha"?
    - Criando uma regra para cada casa vizinha
    - Não há como descrever concisamente um ambiente com muitos objetos
  - Como representar "todo homem é mortal", sem enumerar cada homem e ligá-lo à mortalidade?
  - Lógica proposicional é, então, bastante limitada

Lógica Proposicional

Em suma...

- Desvantagens:
  - Como representar conceitos, quantidades, coisas e pessoas?
  - Como representar "casa vizinha à minha"?
    - Criando uma regra para cada casa vizinha
    - Não há como descrever concisamente um ambiente com muitos objetos
  - Como representar "todo homem é mortal", sem enumerar cada homem e ligá-lo à mortalidade?
  - Lógica proposicional é, então, bastante limitada
  - Precisamos de **Lógica de Primeira Ordem**

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/`
- `http://ocw.mit.edu/OcwWeb/Electrical-Engineering-and-Computer-Science/6-034Spring-2005/LectureNotes/index.htm`
- Suppes, P. (1957): Introduction to Logic. Van Nostrand Reinhold Co.

Referências

- https://en.wikipedia.org/wiki/Monotonicity_of_entailment
- https://www.springer.com/cda/content/document/cda_downloaddocument/9780857292988-c2.pdf
- https://en.wikipedia.org/wiki/Horn_clause