

Um algoritmo baseado em programação dinâmica e renomeamento para minimização de formas normais

Matheus Pimenta

Universidade de Brasília

2016

Conteúdo

- 1 Introdução
- 2 Referencial teórico
- 3 O algoritmo
- 4 Resultados experimentais
- 5 Conclusão
- 6 Referências

Conteúdo

- 1 Introdução
- 2 Referencial teórico
- 3 O algoritmo
- 4 Resultados experimentais
- 5 Conclusão
- 6 Referências

Lógica

- Lógicas são utilizadas para representar e raciocinar sobre problemas computacionais.

Lógica

- Lógicas são utilizadas para representar e raciocinar sobre problemas computacionais.
- A representação se dá através de uma linguagem formal, de *fórmulas*.

Lógica

- Lógicas são utilizadas para representar e raciocinar sobre problemas computacionais.
- A representação se dá através de uma linguagem formal, de *fórmulas*.
- Para atribuir um significado a cada fórmula, define-se para a lógica uma *semântica*

Lógica

- Lógicas são utilizadas para representar e raciocinar sobre problemas computacionais.
- A representação se dá através de uma linguagem formal, de *fórmulas*.
- Para atribuir um significado a cada fórmula, define-se para a lógica uma *semântica*, que possui diferentes *interpretações*.

Lógica

- Lógicas são utilizadas para representar e raciocinar sobre problemas computacionais.
- A representação se dá através de uma linguagem formal, de *fórmulas*.
- Para atribuir um significado a cada fórmula, define-se para a lógica uma *semântica*, que possui diferentes *interpretações*.
- Em lógicas clássicas, os significados possíveis são somente *verdadeiro* ou *falso*.

SAT

Satisfatibilidade: Determinar se existe uma interpretação sob a qual uma dada fórmula é verdadeira.

SAT

Satisfatibilidade: Determinar se existe uma interpretação sob a qual uma dada fórmula é verdadeira.

- Possui grande interesse prático:

SAT

Satisfatibilidade: Determinar se existe uma interpretação sob a qual uma dada fórmula é verdadeira.

- Possui grande interesse prático:
 - Síntese [1], otimização [2] e verificação [3] de *hardware*.

SAT

Satisfatibilidade: Determinar se existe uma interpretação sob a qual uma dada fórmula é verdadeira.

- Possui grande interesse prático:
 - Síntese [1], otimização [2] e verificação [3] de *hardware*.
 - Raciocínio automático [4].

SAT

Satisfatibilidade: Determinar se existe uma interpretação sob a qual uma dada fórmula é verdadeira.

- Possui grande interesse prático:
 - Síntese [1], otimização [2] e verificação [3] de *hardware*.
 - Raciocínio automático [4].
 - Biologia e medicina [5].

SAT

Satisfatibilidade: Determinar se existe uma interpretação sob a qual uma dada fórmula é verdadeira.

- Possui grande interesse prático:
 - Síntese [1], otimização [2] e verificação [3] de *hardware*.
 - Raciocínio automático [4].
 - Biologia e medicina [5].
- Interesse teórico fundamental:

SAT

Satisfatibilidade: Determinar se existe uma interpretação sob a qual uma dada fórmula é verdadeira.

- Possui grande interesse prático:
 - Síntese [1], otimização [2] e verificação [3] de *hardware*.
 - Raciocínio automático [4].
 - Biologia e medicina [5].
- Interesse teórico fundamental:
 - Primeiro problema NP-completo [6].

SAT

Satisfatibilidade: Determinar se existe uma interpretação sob a qual uma dada fórmula é verdadeira.

- Possui grande interesse prático:
 - Síntese [1], otimização [2] e verificação [3] de *hardware*.
 - Raciocínio automático [4].
 - Biologia e medicina [5].
- Interesse teórico fundamental:
 - Primeiro problema NP-completo [6].
 - Deu base para formalizar P *versus* NP [6].

VAL

Validade: Determinar se uma dada fórmula é verdadeira sob qualquer interpretação.

VAL

Validade: Determinar se uma dada fórmula é verdadeira sob qualquer interpretação.

SAT e VAL são redutíveis um ao outro!

Algoritmos para SAT e VAL

- Há diversos algoritmos de busca para SAT e VAL [7, 8, 9].

Algoritmos para SAT e VAL

- Há diversos algoritmos de busca para SAT e VAL [7, 8, 9].
- Conjectura-se que todos são exponenciais [6].

Algoritmos para SAT e VAL

- Há diversos algoritmos de busca para SAT e VAL [7, 8, 9].
- Conjectura-se que todos são exponenciais [6].
- Muitos são baseados em *formas normais*: subconjuntos de fórmulas.

Algoritmos para SAT e VAL

- Há diversos algoritmos de busca para SAT e VAL [7, 8, 9].
- Conjectura-se que todos são exponenciais [6].
- Muitos são baseados em *formas normais*: subconjuntos de fórmulas.
- Algoritmos baseados em formas normais precisam de pré-processamento eficiente.

O trabalho

Hipótese

Considerando melhorar a eficiência total de pré-processamento e busca: fórmulas menores produzem respostas mais rápido?

O trabalho

Hipótese

Considerando melhorar a eficiência total de pré-processamento e busca: fórmulas menores produzem respostas mais rápido?

Objetivo

Testar a hipótese experimentalmente.

O trabalho

- Investigamos algoritmos baseados na *forma normal clausal*.

O trabalho

- Investigamos algoritmos baseados na *forma normal clausal*.
- Tentamos obter fórmulas pequenas reduzindo o *número de cláusulas*

O trabalho

- Investigamos algoritmos baseados na *forma normal clausal*.
- Tentamos obter fórmulas pequenas reduzindo o *número de cláusulas*, através de *renomeamento*.

O trabalho

- Investigamos algoritmos baseados na *forma normal clausal*.
- Tentamos obter fórmulas pequenas reduzindo o *número de cláusulas*, através de *renomeamento*.
- Boy de la Tour [10] e Jackson et al. [11] propõem algoritmos para este problema.

O trabalho

- Investigamos algoritmos baseados na *forma normal clausal*.
- Tentamos obter fórmulas pequenas reduzindo o *número de cláusulas*, através de *renomeamento*.
- Boy de la Tour [10] e Jackson et al. [11] propõem algoritmos para este problema.
- Propomos um algoritmo baseado em programação dinâmica para este problema.

O trabalho

- Investigamos algoritmos baseados na *forma normal clausal*.
- Tentamos obter fórmulas pequenas reduzindo o *número de cláusulas*, através de *renomeamento*.
- Boy de la Tour [10] e Jackson et al. [11] propõem algoritmos para este problema.
- Propomos um algoritmo baseado em programação dinâmica para este problema.
- Comparamos experimentalmente o algoritmo que propomos com o de Boy de la Tour.

Conteúdo

- 1 Introdução
- 2 Referencial teórico
- 3 O algoritmo
- 4 Resultados experimentais
- 5 Conclusão
- 6 Referências

Lógica proposicional

Sintaxe

Símbolos proposicionais

$\mathcal{P} = \{a, b, \dots, a_1, a_2, \dots, b_1, b_2, \dots\}$ é dito o conjunto de *símbolos proposicionais*.

Lógica proposicional

Sintaxe

Símbolos proposicionais

$\mathcal{P} = \{a, b, \dots, a_1, a_2, \dots, b_1, b_2, \dots\}$ é dito o conjunto de *símbolos proposicionais*.

Fórmulas

Se $\phi \in \mathcal{P}$, então ϕ é uma *fórmula*. Além disso, se ϕ_1, \dots, ϕ_n , $n \in \mathbb{N} \cup \{0\}$, são fórmulas, então também são:

Lógica proposicional

Sintaxe

Símbolos proposicionais

$\mathcal{P} = \{a, b, \dots, a_1, a_2, \dots, b_1, b_2, \dots\}$ é dito o conjunto de *símbolos proposicionais*.

Fórmulas

Se $\phi \in \mathcal{P}$, então ϕ é uma *fórmula*. Além disso, se ϕ_1, \dots, ϕ_n , $n \in \mathbb{N} \cup \{0\}$, são fórmulas, então também são:

- 1 *Negação*: $\neg\phi_1$

Lógica proposicional

Sintaxe

Símbolos proposicionais

$\mathcal{P} = \{a, b, \dots, a_1, a_2, \dots, b_1, b_2, \dots\}$ é dito o conjunto de *símbolos proposicionais*.

Fórmulas

Se $\phi \in \mathcal{P}$, então ϕ é uma *fórmula*. Além disso, se ϕ_1, \dots, ϕ_n , $n \in \mathbb{N} \cup \{0\}$, são fórmulas, então também são:

- 1 *Negação*: $\neg\phi_1$
- 2 *Conjunção*: $\phi_1 \wedge \dots \wedge \phi_n$

Lógica proposicional

Sintaxe

Símbolos proposicionais

$\mathcal{P} = \{a, b, \dots, a_1, a_2, \dots, b_1, b_2, \dots\}$ é dito o conjunto de *símbolos proposicionais*.

Fórmulas

Se $\phi \in \mathcal{P}$, então ϕ é uma *fórmula*. Além disso, se ϕ_1, \dots, ϕ_n , $n \in \mathbb{N} \cup \{0\}$, são fórmulas, então também são:

- 1 *Negação*: $\neg\phi_1$
- 2 *Conjunção*: $\phi_1 \wedge \dots \wedge \phi_n$
- 3 *Disjunção*: $\phi_1 \vee \dots \vee \phi_n$

Lógica proposicional

Sintaxe

Símbolos proposicionais

$\mathcal{P} = \{a, b, \dots, a_1, a_2, \dots, b_1, b_2, \dots\}$ é dito o conjunto de *símbolos proposicionais*.

Fórmulas

Se $\phi \in \mathcal{P}$, então ϕ é uma *fórmula*. Além disso, se ϕ_1, \dots, ϕ_n , $n \in \mathbb{N} \cup \{0\}$, são fórmulas, então também são:

- 1 *Negação*: $\neg\phi_1$
- 2 *Conjunção*: $\phi_1 \wedge \dots \wedge \phi_n$
- 3 *Disjunção*: $\phi_1 \vee \dots \vee \phi_n$
- 4 *Implicação*: $\phi_1 \rightarrow \phi_2$

Lógica proposicional

Sintaxe

Símbolos proposicionais

$\mathcal{P} = \{a, b, \dots, a_1, a_2, \dots, b_1, b_2, \dots\}$ é dito o conjunto de *símbolos proposicionais*.

Fórmulas

Se $\phi \in \mathcal{P}$, então ϕ é uma *fórmula*. Além disso, se ϕ_1, \dots, ϕ_n , $n \in \mathbb{N} \cup \{0\}$, são fórmulas, então também são:

- 1 *Negação*: $\neg\phi_1$
- 2 *Conjunção*: $\phi_1 \wedge \dots \wedge \phi_n$
- 3 *Disjunção*: $\phi_1 \vee \dots \vee \phi_n$
- 4 *Implicação*: $\phi_1 \rightarrow \phi_2$
- 5 *Equivalência*: $\phi_1 \leftrightarrow \phi_2$

Lógica proposicional

Sintaxe

Símbolos proposicionais

$\mathcal{P} = \{a, b, \dots, a_1, a_2, \dots, b_1, b_2, \dots\}$ é dito o conjunto de *símbolos proposicionais*.

Fórmulas

Se $\phi \in \mathcal{P}$, então ϕ é uma *fórmula*. Além disso, se ϕ_1, \dots, ϕ_n , $n \in \mathbb{N} \cup \{0\}$, são fórmulas, então também são:

- 1 *Negação*: $\neg\phi_1$
- 2 *Conjunção*: $\phi_1 \wedge \dots \wedge \phi_n$
- 3 *Disjunção*: $\phi_1 \vee \dots \vee \phi_n$
- 4 *Implicação*: $\phi_1 \rightarrow \phi_2$
- 5 *Equivalência*: $\phi_1 \leftrightarrow \phi_2$

Denotamos o conjunto de fórmulas por \mathcal{L} .

Lógica proposicional

Sintaxe – Exemplos

Lógica proposicional

Sintaxe – Exemplos

- $\phi = (p \rightarrow q) \rightarrow \neg s$

Lógica proposicional

Sintaxe – Exemplos

- $\phi = (p \rightarrow q) \rightarrow \neg s$
- $\psi = (p \vee q) \leftrightarrow (r \wedge s)$

Lógica proposicional

Sintaxe – Exemplos

- $\phi = (p \rightarrow q) \rightarrow \neg s$
- $\psi = (p \vee q) \leftrightarrow (r \wedge s)$
- $\xi = \neg(p \rightarrow q)$

Lógica proposicional

Sintaxe

Subfórmulas imediatas

Na definição anterior, as fórmulas ϕ_i são *subfórmulas imediatas*.

Lógica proposicional

Sintaxe

Subfórmulas imediatas

Na definição anterior, as fórmulas ϕ_i são *subfórmulas imediatas*.

Subfórmulas

Dizemos que ψ é subfórmula de ϕ se ψ é subfórmula imediata de ϕ , ou se ψ é subfórmula de ξ e ξ é subfórmula imediata de ϕ .

Lógica proposicional

Sintaxe

Subfórmulas imediatas

Na definição anterior, as fórmulas ϕ_i são *subfórmulas imediatas*.

Subfórmulas

Dizemos que ψ é subfórmula de ϕ se ψ é subfórmula imediata de ϕ , ou se ψ é subfórmula de ξ e ξ é subfórmula imediata de ϕ .

Notação: $\psi \sqsubset \phi$ e $\{\psi \mid \psi \sqsubset \phi\} = SF(\phi)$

Lógica proposicional

Sintaxe

Subfórmulas imediatas

Na definição anterior, as fórmulas ϕ_i são *subfórmulas imediatas*.

Subfórmulas

Dizemos que ψ é subfórmula de ϕ se ψ é subfórmula imediata de ϕ , ou se ψ é subfórmula de ξ e ξ é subfórmula imediata de ϕ .

Notação: $\psi \sqsubset \phi$ e $\{\psi \mid \psi \sqsubset \phi\} = SF(\phi)$

Exemplo: $\phi = (p \wedge q \wedge (r \rightarrow s))$

Lógica proposicional

Sintaxe

Subfórmulas imediatas

Na definição anterior, as fórmulas ϕ_i são *subfórmulas imediatas*.

Subfórmulas

Dizemos que ψ é subfórmula de ϕ se ψ é subfórmula imediata de ϕ , ou se ψ é subfórmula de ξ e ξ é subfórmula imediata de ϕ .

Notação: $\psi \sqsubset \phi$ e $\{\psi \mid \psi \sqsubset \phi\} = SF(\phi)$

Exemplo: $\phi = (p \wedge q \wedge (r \rightarrow s))$

p , q e $r \rightarrow s$ são subfórmulas imediatas de ϕ .

Lógica proposicional

Sintaxe

Subfórmulas imediatas

Na definição anterior, as fórmulas ϕ_i são *subfórmulas imediatas*.

Subfórmulas

Dizemos que ψ é subfórmula de ϕ se ψ é subfórmula imediata de ϕ , ou se ψ é subfórmula de ξ e ξ é subfórmula imediata de ϕ .

Notação: $\psi \sqsubset \phi$ e $\{\psi \mid \psi \sqsubset \phi\} = SF(\phi)$

Exemplo: $\phi = (p \wedge q \wedge (r \rightarrow s))$

p , q e $r \rightarrow s$ são subfórmulas imediatas de ϕ .

$SF(\phi) = \{p, q, r \rightarrow s, r, s\}$

Lógica proposicional

Semântica

Valorações booleanas

Dizemos que v_0 é uma *valoração booleana* se $v_0 : \mathcal{P} \mapsto \{V, F\}$.

Lógica proposicional

Semântica

Valorações booleanas

Dizemos que \mathbb{v}_0 é uma *valoração booleana* se $\mathbb{v}_0 : \mathcal{P} \mapsto \{V, F\}$.

Interpretações

Seja \mathbb{v}_0 é uma valoração booleana. Dizemos que $\mathbb{v} : \mathcal{L} \mapsto \{V, F\}$ é uma *interpretação definida por \mathbb{v}_0* , se:

Lógica proposicional

Semântica

Valorações booleanas

Dizemos que v_0 é uma *valoração booleana* se $v_0 : \mathcal{P} \mapsto \{V, F\}$.

Interpretações

Seja v_0 é uma valoração booleana. Dizemos que $v : \mathcal{L} \mapsto \{V, F\}$ é uma *interpretação definida por* v_0 , se:

- 1 Se $\phi_1 \in \mathcal{P}$, então $v(\phi_1) = v_0(\phi_1)$.

Lógica proposicional

Semântica

Valorações booleanas

Dizemos que v_0 é uma *valoração booleana* se $v_0 : \mathcal{P} \mapsto \{V, F\}$.

Interpretações

Seja v_0 é uma valoração booleana. Dizemos que $v : \mathcal{L} \mapsto \{V, F\}$ é uma *interpretação definida por v_0* , se:

- 1 Se $\phi_1 \in \mathcal{P}$, então $v(\phi_1) = v_0(\phi_1)$.
- 2 $v(\neg\phi_1) = V$ se, e somente se, $v(\phi_1) = F$.

Lógica proposicional

Semântica

Valorações booleanas

Dizemos que v_0 é uma *valoração booleana* se $v_0 : \mathcal{P} \mapsto \{V, F\}$.

Interpretações

Seja v_0 é uma valoração booleana. Dizemos que $v : \mathcal{L} \mapsto \{V, F\}$ é uma *interpretação definida por* v_0 , se:

- 1 Se $\phi_1 \in \mathcal{P}$, então $v(\phi_1) = v_0(\phi_1)$.
- 2 $v(\neg\phi_1) = V$ se, e somente se, $v(\phi_1) = F$.
- 3 $v(\phi_1 \wedge \dots \wedge \phi_n) = V$ se, e somente se, $v(\phi_i) = V$, para todo i .

Lógica proposicional

Semântica

Valorações booleanas

Dizemos que v_0 é uma *valoração booleana* se $v_0 : \mathcal{P} \mapsto \{V, F\}$.

Interpretações

Seja v_0 é uma valoração booleana. Dizemos que $v : \mathcal{L} \mapsto \{V, F\}$ é uma *interpretação definida por* v_0 , se:

- 1 Se $\phi_1 \in \mathcal{P}$, então $v(\phi_1) = v_0(\phi_1)$.
- 2 $v(\neg\phi_1) = V$ se, e somente se, $v(\phi_1) = F$.
- 3 $v(\phi_1 \wedge \dots \wedge \phi_n) = V$ se, e somente se, $v(\phi_i) = V$, para todo i .
- 4 $v(\phi_1 \vee \dots \vee \phi_n) = V$ se, e somente se, $v(\phi_i) = V$, para algum i .

Lógica proposicional

Semântica

Valorações booleanas

Dizemos que v_0 é uma *valoração booleana* se $v_0 : \mathcal{P} \mapsto \{V, F\}$.

Interpretações

Seja v_0 é uma valoração booleana. Dizemos que $v : \mathcal{L} \mapsto \{V, F\}$ é uma *interpretação definida por* v_0 , se:

- 1 Se $\phi_1 \in \mathcal{P}$, então $v(\phi_1) = v_0(\phi_1)$.
- 2 $v(\neg\phi_1) = V$ se, e somente se, $v(\phi_1) = F$.
- 3 $v(\phi_1 \wedge \dots \wedge \phi_n) = V$ se, e somente se, $v(\phi_i) = V$, para todo i .
- 4 $v(\phi_1 \vee \dots \vee \phi_n) = V$ se, e somente se, $v(\phi_i) = V$, para algum i .
- 5 $v(\phi_1 \rightarrow \phi_2) = V$ se, e somente se, $v(\phi_1) = F$ ou $v(\phi_2) = V$.

Lógica proposicional

Semântica

Valorações booleanas

Dizemos que v_0 é uma *valoração booleana* se $v_0 : \mathcal{P} \mapsto \{V, F\}$.

Interpretações

Seja v_0 é uma valoração booleana. Dizemos que $v : \mathcal{L} \mapsto \{V, F\}$ é uma *interpretação definida por* v_0 , se:

- 1 Se $\phi_1 \in \mathcal{P}$, então $v(\phi_1) = v_0(\phi_1)$.
- 2 $v(\neg\phi_1) = V$ se, e somente se, $v(\phi_1) = F$.
- 3 $v(\phi_1 \wedge \dots \wedge \phi_n) = V$ se, e somente se, $v(\phi_i) = V$, para todo i .
- 4 $v(\phi_1 \vee \dots \vee \phi_n) = V$ se, e somente se, $v(\phi_i) = V$, para algum i .
- 5 $v(\phi_1 \rightarrow \phi_2) = V$ se, e somente se, $v(\phi_1) = F$ ou $v(\phi_2) = V$.
- 6 $v(\phi_1 \leftrightarrow \phi_2) = V$ se, e somente se, $v(\phi_1) = v(\phi_2)$.

Lógica proposicional

Semântica – Exemplos

Seja \mathbb{V} definida por $\mathbb{V}_0 = \{(p, V), (q, F), (r, V), (s, V)\}$ e considere $\phi = \neg((p \vee (q \wedge r \wedge s)) \leftrightarrow (q \rightarrow \neg s))$. Então:

Lógica proposicional

Semântica – Exemplos

Seja \mathbb{V} definida por $\mathbb{V}_0 = \{(p, V), (q, F), (r, V), (s, V)\}$ e considere $\phi = \neg((p \vee (q \wedge r \wedge s)) \leftrightarrow (q \rightarrow \neg s))$. Então:

① $\mathbb{V}(q \wedge r \wedge s) = F$

Lógica proposicional

Semântica – Exemplos

Seja \mathbb{v} definida por $\mathbb{v}_0 = \{(p, V), (q, F), (r, V), (s, V)\}$ e considere $\phi = \neg((p \vee (q \wedge r \wedge s)) \leftrightarrow (q \rightarrow \neg s))$. Então:

- ① $\mathbb{v}(q \wedge r \wedge s) = F$
- ② $\mathbb{v}(p \vee (q \wedge r \wedge s)) = V$

Lógica proposicional

Semântica – Exemplos

Seja \mathbb{V} definida por $\mathbb{V}_0 = \{(p, V), (q, F), (r, V), (s, V)\}$ e considere $\phi = \neg((p \vee (q \wedge r \wedge s)) \leftrightarrow (q \rightarrow \neg s))$. Então:

- 1 $\mathbb{V}(q \wedge r \wedge s) = F$
- 2 $\mathbb{V}(p \vee (q \wedge r \wedge s)) = V$
- 3 $\mathbb{V}(\neg s) = F$

Lógica proposicional

Semântica – Exemplos

Seja \mathbb{V} definida por $\mathbb{V}_0 = \{(p, V), (q, F), (r, V), (s, V)\}$ e considere $\phi = \neg((p \vee (q \wedge r \wedge s)) \leftrightarrow (q \rightarrow \neg s))$. Então:

- 1 $\mathbb{V}(q \wedge r \wedge s) = F$
- 2 $\mathbb{V}(p \vee (q \wedge r \wedge s)) = V$
- 3 $\mathbb{V}(\neg s) = F$
- 4 $\mathbb{V}(q \rightarrow \neg s) = V$

Lógica proposicional

Semântica – Exemplos

Seja \mathbb{v} definida por $\mathbb{v}_0 = \{(p, V), (q, F), (r, V), (s, V)\}$ e considere $\phi = \neg((p \vee (q \wedge r \wedge s)) \leftrightarrow (q \rightarrow \neg s))$. Então:

① $\mathbb{v}(q \wedge r \wedge s) = F$

② $\mathbb{v}(p \vee (q \wedge r \wedge s)) = V$

③ $\mathbb{v}(\neg s) = F$

④ $\mathbb{v}(q \rightarrow \neg s) = V$

⑤ $\mathbb{v}((p \vee (q \wedge r \wedge s)) \leftrightarrow (q \rightarrow \neg s)) = V$

Lógica proposicional

Semântica – Exemplos

Seja v definida por $v_0 = \{(p, V), (q, F), (r, V), (s, V)\}$ e considere $\phi = \neg((p \vee (q \wedge r \wedge s)) \leftrightarrow (q \rightarrow \neg s))$. Então:

① $v(q \wedge r \wedge s) = F$

② $v(p \vee (q \wedge r \wedge s)) = V$

③ $v(\neg s) = F$

④ $v(q \rightarrow \neg s) = V$

⑤ $v((p \vee (q \wedge r \wedge s)) \leftrightarrow (q \rightarrow \neg s)) = V$

⑥ $v(\phi) = F$

Lógica proposicional

Semântica – Algumas definições

Lógica proposicional

Semântica – Algumas definições

- 1 Se existe v tal que $v(\phi) = V$, dizemos que ϕ é *satisfatível*.

Lógica proposicional

Semântica – Algumas definições

- 1 Se existe v tal que $v(\phi) = V$, dizemos que ϕ é *satisfatível*.
- 2 Se existe v tal que $v(\phi) = F$, dizemos que ϕ é *falsificável*.

Lógica proposicional

Semântica – Algumas definições

- 1 Se existe v tal que $v(\phi) = V$, dizemos que ϕ é *satisfatível*.
- 2 Se existe v tal que $v(\phi) = F$, dizemos que ϕ é *falsificável*.
- 3 Se $v(\phi) = V$ para toda v , dizemos que ϕ é uma *tautologia*.

Lógica proposicional

Semântica – Algumas definições

- 1 Se existe v tal que $v(\phi) = V$, dizemos que ϕ é *satisfatível*.
- 2 Se existe v tal que $v(\phi) = F$, dizemos que ϕ é *falsificável*.
- 3 Se $v(\phi) = V$ para toda v , dizemos que ϕ é uma *tautologia*.
- 4 Se $v(\phi) = F$ para toda v , dizemos que ϕ é uma *contradição*, ou que ϕ é *insatisfatível*.

Lógica proposicional

Semântica – Algumas definições

- 1 Se existe v tal que $v(\phi) = V$, dizemos que ϕ é *satisfatível*.
- 2 Se existe v tal que $v(\phi) = F$, dizemos que ϕ é *falsificável*.
- 3 Se $v(\phi) = V$ para toda v , dizemos que ϕ é uma *tautologia*.
- 4 Se $v(\phi) = F$ para toda v , dizemos que ϕ é uma *contradição*, ou que ϕ é *insatisfatível*.
- 5 Se ϕ é satisfatível e falsificável, dizemos que ϕ é uma *contingência*.

Lógica proposicional

Semântica – Exemplos

Seja ϕ uma fórmula qualquer, ψ_1 uma tautologia, ψ_2 uma contradição e ψ_3 uma contingência. Então:

Lógica proposicional

Semântica – Exemplos

Seja ϕ uma fórmula qualquer, ψ_1 uma tautologia, ψ_2 uma contradição e ψ_3 uma contingência. Então:

São tautologias:

Lógica proposicional

Semântica – Exemplos

Seja ϕ uma fórmula qualquer, ψ_1 uma tautologia, ψ_2 uma contradição e ψ_3 uma contingência. Então:

São tautologias:

- $\phi \vee \neg\phi$

Lógica proposicional

Semântica – Exemplos

Seja ϕ uma fórmula qualquer, ψ_1 uma tautologia, ψ_2 uma contradição e ψ_3 uma contingência. Então:

São tautologias:

- $\phi \vee \neg\phi$
- $\phi \rightarrow \phi$

Lógica proposicional

Semântica – Exemplos

Seja ϕ uma fórmula qualquer, ψ_1 uma tautologia, ψ_2 uma contradição e ψ_3 uma contingência. Então:

São tautologias:

- $\phi \vee \neg\phi$
- $\phi \rightarrow \phi$
- $\phi \leftrightarrow \phi$

Lógica proposicional

Semântica – Exemplos

Seja ϕ uma fórmula qualquer, ψ_1 uma tautologia, ψ_2 uma contradição e ψ_3 uma contingência. Então:

São tautologias:

- $\phi \vee \neg\phi$
- $\phi \rightarrow \phi$
- $\phi \leftrightarrow \phi$
- $\neg\psi_2$

Lógica proposicional

Semântica – Exemplos

Seja ϕ uma fórmula qualquer, ψ_1 uma tautologia, ψ_2 uma contradição e ψ_3 uma contingência. Então:

São tautologias:

- $\phi \vee \neg\phi$
- $\phi \rightarrow \phi$
- $\phi \leftrightarrow \phi$
- $\neg\psi_2$

São contradições:

Lógica proposicional

Semântica – Exemplos

Seja ϕ uma fórmula qualquer, ψ_1 uma tautologia, ψ_2 uma contradição e ψ_3 uma contingência. Então:

São tautologias:

- $\phi \vee \neg\phi$
- $\phi \rightarrow \phi$
- $\phi \leftrightarrow \phi$
- $\neg\psi_2$

São contradições:

- $\phi \wedge \neg\phi$

Lógica proposicional

Semântica – Exemplos

Seja ϕ uma fórmula qualquer, ψ_1 uma tautologia, ψ_2 uma contradição e ψ_3 uma contingência. Então:

São tautologias:

- $\phi \vee \neg\phi$
- $\phi \rightarrow \phi$
- $\phi \leftrightarrow \phi$
- $\neg\psi_2$

São contradições:

- $\phi \wedge \neg\phi$
- $\phi \leftrightarrow \neg\phi$

Lógica proposicional

Semântica – Exemplos

Seja ϕ uma fórmula qualquer, ψ_1 uma tautologia, ψ_2 uma contradição e ψ_3 uma contingência. Então:

São tautologias:

- $\phi \vee \neg\phi$
- $\phi \rightarrow \phi$
- $\phi \leftrightarrow \phi$
- $\neg\psi_2$

São contradições:

- $\phi \wedge \neg\phi$
- $\phi \leftrightarrow \neg\phi$
- $\neg\psi_1$

Lógica proposicional

Semântica – Exemplos

Seja ϕ uma fórmula qualquer, ψ_1 uma tautologia, ψ_2 uma contradição e ψ_3 uma contingência. Então:

São tautologias:

- $\phi \vee \neg\phi$
- $\phi \rightarrow \phi$
- $\phi \leftrightarrow \phi$
- $\neg\psi_2$

São contradições:

- $\phi \wedge \neg\phi$
- $\phi \leftrightarrow \neg\phi$
- $\neg\psi_1$

São contingências:

Lógica proposicional

Semântica – Exemplos

Seja ϕ uma fórmula qualquer, ψ_1 uma tautologia, ψ_2 uma contradição e ψ_3 uma contingência. Então:

São tautologias:

- $\phi \vee \neg\phi$
- $\phi \rightarrow \phi$
- $\phi \leftrightarrow \phi$
- $\neg\psi_2$

São contradições:

- $\phi \wedge \neg\phi$
- $\phi \leftrightarrow \neg\phi$
- $\neg\psi_1$

São contingências:

- p

Lógica proposicional

Semântica – Exemplos

Seja ϕ uma fórmula qualquer, ψ_1 uma tautologia, ψ_2 uma contradição e ψ_3 uma contingência. Então:

São tautologias:

- $\phi \vee \neg\phi$
- $\phi \rightarrow \phi$
- $\phi \leftrightarrow \phi$
- $\neg\psi_2$

São contingências:

- p
- $\neg p$

São contradições:

- $\phi \wedge \neg\phi$
- $\phi \leftrightarrow \neg\phi$
- $\neg\psi_1$

Lógica proposicional

Semântica – Exemplos

Seja ϕ uma fórmula qualquer, ψ_1 uma tautologia, ψ_2 uma contradição e ψ_3 uma contingência. Então:

São tautologias:

- $\phi \vee \neg\phi$
- $\phi \rightarrow \phi$
- $\phi \leftrightarrow \phi$
- $\neg\psi_2$

São contradições:

- $\phi \wedge \neg\phi$
- $\phi \leftrightarrow \neg\phi$
- $\neg\psi_1$

São contingências:

- p
- $\neg p$
- $p \wedge q$

Lógica proposicional

Semântica – Exemplos

Seja ϕ uma fórmula qualquer, ψ_1 uma tautologia, ψ_2 uma contradição e ψ_3 uma contingência. Então:

São tautologias:

- $\phi \vee \neg\phi$
- $\phi \rightarrow \phi$
- $\phi \leftrightarrow \phi$
- $\neg\psi_2$

São contradições:

- $\phi \wedge \neg\phi$
- $\phi \leftrightarrow \neg\phi$
- $\neg\psi_1$

São contingências:

- p
- $\neg p$
- $p \wedge q$
- $p \vee q$

Lógica proposicional

Semântica – Exemplos

Seja ϕ uma fórmula qualquer, ψ_1 uma tautologia, ψ_2 uma contradição e ψ_3 uma contingência. Então:

São tautologias:

- $\phi \vee \neg\phi$
- $\phi \rightarrow \phi$
- $\phi \leftrightarrow \phi$
- $\neg\psi_2$

São contradições:

- $\phi \wedge \neg\phi$
- $\phi \leftrightarrow \neg\phi$
- $\neg\psi_1$

São contingências:

- p
- $\neg p$
- $p \wedge q$
- $p \vee q$
- $p \rightarrow q$

Lógica proposicional

Semântica – Exemplos

Seja ϕ uma fórmula qualquer, ψ_1 uma tautologia, ψ_2 uma contradição e ψ_3 uma contingência. Então:

São tautologias:

- $\phi \vee \neg\phi$
- $\phi \rightarrow \phi$
- $\phi \leftrightarrow \phi$
- $\neg\psi_2$

São contradições:

- $\phi \wedge \neg\phi$
- $\phi \leftrightarrow \neg\phi$
- $\neg\psi_1$

São contingências:

- p
- $\neg p$
- $p \wedge q$
- $p \vee q$
- $p \rightarrow q$
- $p \leftrightarrow q$

Lógica proposicional

Semântica – Exemplos

Seja ϕ uma fórmula qualquer, ψ_1 uma tautologia, ψ_2 uma contradição e ψ_3 uma contingência. Então:

São tautologias:

- $\phi \vee \neg\phi$
- $\phi \rightarrow \phi$
- $\phi \leftrightarrow \phi$
- $\neg\psi_2$

São contradições:

- $\phi \wedge \neg\phi$
- $\phi \leftrightarrow \neg\phi$
- $\neg\psi_1$

São contingências:

- p
- $\neg p$
- $p \wedge q$
- $p \vee q$
- $p \rightarrow q$
- $p \leftrightarrow q$
- $\neg\psi_3$

Problemas da lógica proposicional

Seja $L \subseteq \mathcal{L}$. Se nos referimos a L como um *problema*, referimo-nos ao problema de, dada ϕ qualquer, determinar se $\phi \in L$ ou se $\phi \notin L$.

Problemas da lógica proposicional

Seja $L \subseteq \mathcal{L}$. Se nos referimos a L como um *problema*, referimo-nos ao problema de, dada ϕ qualquer, determinar se $\phi \in L$ ou se $\phi \notin L$.

❶ $\text{SAT} = \{\phi \in \mathcal{L} \mid \phi \text{ é satisfatível}\}$

Problemas da lógica proposicional

Seja $L \subseteq \mathcal{L}$. Se nos referimos a L como um *problema*, referimo-nos ao problema de, dada ϕ qualquer, determinar se $\phi \in L$ ou se $\phi \notin L$.

- 1 SAT = $\{\phi \in \mathcal{L} \mid \phi \text{ é satisfatível}\}$
- 2 UNSAT = $\{\phi \in \mathcal{L} \mid \phi \text{ é insatisfatível}\}$

Problemas da lógica proposicional

Seja $L \subseteq \mathcal{L}$. Se nos referimos a L como um *problema*, referimo-nos ao problema de, dada ϕ qualquer, determinar se $\phi \in L$ ou se $\phi \notin L$.

- 1 $\text{SAT} = \{\phi \in \mathcal{L} \mid \phi \text{ é satisfatível}\}$
- 2 $\text{UNSAT} = \{\phi \in \mathcal{L} \mid \phi \text{ é insatisfatível}\} = \overline{\text{SAT}}$

Problemas da lógica proposicional

Seja $L \subseteq \mathcal{L}$. Se nos referimos a L como um *problema*, referimo-nos ao problema de, dada ϕ qualquer, determinar se $\phi \in L$ ou se $\phi \notin L$.

- ① $\text{SAT} = \{\phi \in \mathcal{L} \mid \phi \text{ é satisfatível}\}$
- ② $\text{UNSAT} = \{\phi \in \mathcal{L} \mid \phi \text{ é insatisfatível}\} = \overline{\text{SAT}}$
- ③ $\text{VAL} = \{\phi \in \mathcal{L} \mid \phi \text{ é tautologia}\}$

Problemas da lógica proposicional

Seja $L \subseteq \mathcal{L}$. Se nos referimos a L como um *problema*, referimo-nos ao problema de, dada ϕ qualquer, determinar se $\phi \in L$ ou se $\phi \notin L$.

- ① $\text{SAT} = \{\phi \in \mathcal{L} \mid \phi \text{ é satisfatível}\}$
- ② $\text{UNSAT} = \{\phi \in \mathcal{L} \mid \phi \text{ é insatisfatível}\} = \overline{\text{SAT}}$
- ③ $\text{VAL} = \{\phi \in \mathcal{L} \mid \phi \text{ é tautologia}\}$

Observações:

Problemas da lógica proposicional

Seja $L \subseteq \mathcal{L}$. Se nos referimos a L como um *problema*, referimo-nos ao problema de, dada ϕ qualquer, determinar se $\phi \in L$ ou se $\phi \notin L$.

- ❶ $\text{SAT} = \{\phi \in \mathcal{L} \mid \phi \text{ é satisfatível}\}$
- ❷ $\text{UNSAT} = \{\phi \in \mathcal{L} \mid \phi \text{ é insatisfatível}\} = \overline{\text{SAT}}$
- ❸ $\text{VAL} = \{\phi \in \mathcal{L} \mid \phi \text{ é tautologia}\}$

Observações:

- ❶ Há algoritmos para SAT [7, 8, 9].

Problemas da lógica proposicional

Seja $L \subseteq \mathcal{L}$. Se nos referimos a L como um *problema*, referimo-nos ao problema de, dada ϕ qualquer, determinar se $\phi \in L$ ou se $\phi \notin L$.

- ① $\text{SAT} = \{\phi \in \mathcal{L} \mid \phi \text{ é satisfatível}\}$
- ② $\text{UNSAT} = \{\phi \in \mathcal{L} \mid \phi \text{ é insatisfatível}\} = \overline{\text{SAT}}$
- ③ $\text{VAL} = \{\phi \in \mathcal{L} \mid \phi \text{ é tautologia}\}$

Observações:

- ① Há algoritmos para SAT [7, 8, 9].
- ② SAT e UNSAT são redutíveis um ao outro.

Problemas da lógica proposicional

Seja $L \subseteq \mathcal{L}$. Se nos referimos a L como um *problema*, referimo-nos ao problema de, dada ϕ qualquer, determinar se $\phi \in L$ ou se $\phi \notin L$.

- ① $\text{SAT} = \{\phi \in \mathcal{L} \mid \phi \text{ é satisfatível}\}$
- ② $\text{UNSAT} = \{\phi \in \mathcal{L} \mid \phi \text{ é insatisfatível}\} = \overline{\text{SAT}}$
- ③ $\text{VAL} = \{\phi \in \mathcal{L} \mid \phi \text{ é tautologia}\}$

Observações:

- ① Há algoritmos para SAT [7, 8, 9].
- ② SAT e UNSAT são redutíveis um ao outro. (claro!)

Problemas da lógica proposicional

Seja $L \subseteq \mathcal{L}$. Se nos referimos a L como um *problema*, referimo-nos ao problema de, dada ϕ qualquer, determinar se $\phi \in L$ ou se $\phi \notin L$.

- ① $\text{SAT} = \{\phi \in \mathcal{L} \mid \phi \text{ é satisfatível}\}$
- ② $\text{UNSAT} = \{\phi \in \mathcal{L} \mid \phi \text{ é insatisfatível}\} = \overline{\text{SAT}}$
- ③ $\text{VAL} = \{\phi \in \mathcal{L} \mid \phi \text{ é tautologia}\}$

Observações:

- ① Há algoritmos para SAT [7, 8, 9].
- ② SAT e UNSAT são redutíveis um ao outro. (claro!)
- ③ SAT e VAL são redutíveis um ao outro.

Problemas da lógica proposicional

Seja A_{UNSAT} um algoritmo para UNSAT e $R_{\text{VAL}} =$

“Sobre a entrada $\phi \in \mathcal{L}$: Dê a resposta de A_{UNSAT} sobre $\neg\phi$.”

ϕ	$\neg\phi$	$A_{\text{UNSAT}}(\neg\phi)$	$R_{\text{VAL}}(\phi)$
Tautologia	Contradição	Sim	Sim
Contradição	Tautologia	Não	Não
Contingência	Contingência	Não	Não

Problemas da lógica proposicional

Seja A_{UNSAT} um algoritmo para UNSAT e $R_{\text{VAL}} =$

“Sobre a entrada $\phi \in \mathcal{L}$: Dê a resposta de A_{UNSAT} sobre $\neg\phi$.”

ϕ	$\neg\phi$	$A_{\text{UNSAT}}(\neg\phi)$	$R_{\text{VAL}}(\phi)$
Tautologia	Contradição	Sim	Sim
Contradição	Tautologia	Não	Não
Contingência	Contingência	Não	Não

Seja A_{VAL} um algoritmo para VAL e $R_{\text{UNSAT}} =$

“Sobre a entrada $\phi \in \mathcal{L}$: Dê a resposta de A_{VAL} sobre $\neg\phi$.”

ϕ	$\neg\phi$	$A_{\text{VAL}}(\neg\phi)$	$R_{\text{UNSAT}}(\phi)$
Tautologia	Contradição	Não	Não
Contradição	Tautologia	Sim	Sim
Contingência	Contingência	Não	Não

Formas normais

Regras de reescrita

Uma *regra de reescrita* que transforma ϕ em ψ , escrito $\phi \mapsto \psi$,

Formas normais

Regras de reescrita

Uma *regra de reescrita* que transforma ϕ em ψ , escrito $\phi \mapsto \psi$,

① *preserva equivalência* se, e somente se, $\mathbb{V}(\phi) = \mathbb{V}(\psi)$, $\forall \mathbb{V}$.

Formas normais

Regras de reescrita

Uma *regra de reescrita* que transforma ϕ em ψ , escrito $\phi \mapsto \psi$,

- 1 *preserva equivalência* se, e somente se, $\vDash(\phi) = \vDash(\psi)$, $\forall \mathbf{v}$.
- 2 *preserva satisfatibilidade* se, e somente se, $\phi, \psi \in \text{SAT}$ ou $\phi, \psi \notin \text{SAT}$.

Formas normais

Forma normal negada (FNN)

$$(p \wedge \neg q \wedge \neg r) \vee (x \wedge \neg y \wedge (r \vee s))$$

Formas normais

Forma normal negada (FNN)

$$(p \wedge \neg q \wedge \neg r) \vee (x \wedge \neg y \wedge (r \vee s))$$

As transformações:

Formas normais

Forma normal negada (FNN)

$$(p \wedge \neg q \wedge \neg r) \vee (x \wedge \neg y \wedge (r \vee s))$$

As transformações:

① $\neg\neg\phi_1 \longmapsto \phi_1$ (eliminação de dupla negação)

Formas normais

Forma normal negada (FNN)

$$(p \wedge \neg q \wedge \neg r) \vee (x \wedge \neg y \wedge (r \vee s))$$

As transformações:

- 1 $\neg\neg\phi_1 \longmapsto \phi_1$ (eliminação de dupla negação)
- 2 $\neg(\phi_1 \wedge \dots \wedge \phi_n) \longmapsto \neg\phi_1 \vee \dots \vee \neg\phi_n$ (De Morgan)

Formas normais

Forma normal negada (FNN)

$$(p \wedge \neg q \wedge \neg r) \vee (x \wedge \neg y \wedge (r \vee s))$$

As transformações:

- 1 $\neg\neg\phi_1 \longmapsto \phi_1$ (eliminação de dupla negação)
- 2 $\neg(\phi_1 \wedge \dots \wedge \phi_n) \longmapsto \neg\phi_1 \vee \dots \vee \neg\phi_n$ (De Morgan)
- 3 $\neg(\phi_1 \vee \dots \vee \phi_n) \longmapsto \neg\phi_1 \wedge \dots \wedge \neg\phi_n$ (De Morgan)

Formas normais

Forma normal negada (FNN)

$$(p \wedge \neg q \wedge \neg r) \vee (x \wedge \neg y \wedge (r \vee s))$$

As transformações:

- 1 $\neg\neg\phi_1 \mapsto \phi_1$ (eliminação de dupla negação)
- 2 $\neg(\phi_1 \wedge \dots \wedge \phi_n) \mapsto \neg\phi_1 \vee \dots \vee \neg\phi_n$ (De Morgan)
- 3 $\neg(\phi_1 \vee \dots \vee \phi_n) \mapsto \neg\phi_1 \wedge \dots \wedge \neg\phi_n$ (De Morgan)
- 4 $\phi_1 \rightarrow \phi_2 \mapsto \neg\phi_1 \vee \phi_2$

Formas normais

Forma normal negada (FNN)

$$(p \wedge \neg q \wedge \neg r) \vee (x \wedge \neg y \wedge (r \vee s))$$

As transformações:

- 1 $\neg\neg\phi_1 \mapsto \phi_1$ (eliminação de dupla negação)
- 2 $\neg(\phi_1 \wedge \dots \wedge \phi_n) \mapsto \neg\phi_1 \vee \dots \vee \neg\phi_n$ (De Morgan)
- 3 $\neg(\phi_1 \vee \dots \vee \phi_n) \mapsto \neg\phi_1 \wedge \dots \wedge \neg\phi_n$ (De Morgan)
- 4 $\phi_1 \rightarrow \phi_2 \mapsto \neg\phi_1 \vee \phi_2$
- 5 $\phi|_{\pi} \mapsto (\phi_1 \rightarrow \phi_2) \wedge (\phi_2 \rightarrow \phi_1)$

Formas normais

Forma normal negada (FNN)

$$(p \wedge \neg q \wedge \neg r) \vee (x \wedge \neg y \wedge (r \vee s))$$

As transformações:

- 1 $\neg\neg\phi_1 \mapsto \phi_1$ (eliminação de dupla negação)
- 2 $\neg(\phi_1 \wedge \dots \wedge \phi_n) \mapsto \neg\phi_1 \vee \dots \vee \neg\phi_n$ (De Morgan)
- 3 $\neg(\phi_1 \vee \dots \vee \phi_n) \mapsto \neg\phi_1 \wedge \dots \wedge \neg\phi_n$ (De Morgan)
- 4 $\phi_1 \rightarrow \phi_2 \mapsto \neg\phi_1 \vee \phi_2$
- 5 $\phi|_{\pi} \mapsto (\phi_1 \rightarrow \phi_2) \wedge (\phi_2 \rightarrow \phi_1)$

preservam equivalência!

Formas normais

Forma normal clausal (FNC)

$$(p \vee \neg q \vee \neg r) \wedge (x \vee \neg y \vee r \vee s) \wedge (a \vee \neg b \vee c)$$

Formas normais

Forma normal clausal (FNC)

$$(p \vee \neg q \vee \neg r) \wedge (x \vee \neg y \vee r \vee s) \wedge (a \vee \neg b \vee c)$$

A transformação:

$$\phi \vee (\psi \wedge \xi) \longmapsto (\phi \vee \psi) \wedge (\phi \vee \xi) \quad (\text{distribuição})$$

Formas normais

Forma normal clausal (FNC)

$$(p \vee \neg q \vee \neg r) \wedge (x \vee \neg y \vee r \vee s) \wedge (a \vee \neg b \vee c)$$

A transformação:

$$\phi \vee (\psi \wedge \xi) \longmapsto (\phi \vee \psi) \wedge (\phi \vee \xi) \quad (\text{distribuição})$$

preserva equivalência!

Renomeamento

- 1 Escolhemos um conjunto de subfórmulas $R \subseteq SF(\phi)$.

Renomeamento

- 1 Escolhemos um conjunto de subfórmulas $R \subseteq SF(\phi)$.
- 2 Para cada $\psi \in R$:

Renomeamento

- 1 Escolhemos um conjunto de subfórmulas $R \subseteq SF(\phi)$.
- 2 Para cada $\psi \in R$:
 - 1 Escolhemos um símbolo proposicional novo $s(\psi) \in \mathcal{P}$.

Renomeamento

- 1 Escolhemos um conjunto de subfórmulas $R \subseteq SF(\phi)$.
- 2 Para cada $\psi \in R$:
 - 1 Escolhemos um símbolo proposicional novo $s(\psi) \in \mathcal{P}$.
 - 2 Trocamos todas as ocorrências de ψ por $s(\psi)$.

Renomeamento

- 1 Escolhemos um conjunto de subfórmulas $R \subseteq SF(\phi)$.
- 2 Para cada $\psi \in R$:
 - 1 Escolhemos um símbolo proposicional novo $s(\psi) \in \mathcal{P}$.
 - 2 Trocamos todas as ocorrências de ψ por $s(\psi)$.
 - 3 Incluímos a definição $s(\psi) \rightarrow \psi$ em conjunção.

Renomeamento

- 1 Escolhemos um conjunto de subfórmulas $R \subseteq SF(\phi)$.
- 2 Para cada $\psi \in R$:
 - 1 Escolhemos um símbolo proposicional novo $s(\psi) \in \mathcal{P}$.
 - 2 Trocamos todas as ocorrências de ψ por $s(\psi)$.
 - 3 Incluímos a definição $s(\psi) \rightarrow \psi$ em conjunção.

Exemplo: $(\neg p_1 \wedge p_2 \wedge p_3) \vee (\neg q_1 \wedge q_2 \wedge \neg q_3) \vee (r_1 \wedge r_2 \wedge \neg r_3)$

Renomeamento

- 1 Escolhemos um conjunto de subfórmulas $R \subseteq SF(\phi)$.
- 2 Para cada $\psi \in R$:
 - 1 Escolhemos um símbolo proposicional novo $s(\psi) \in \mathcal{P}$.
 - 2 Trocamos todas as ocorrências de ψ por $s(\psi)$.
 - 3 Incluímos a definição $s(\psi) \rightarrow \psi$ em conjunção.

Exemplo: $(\neg p_1 \wedge p_2 \wedge p_3) \vee (\neg q_1 \wedge q_2 \wedge \neg q_3) \vee (r_1 \wedge r_2 \wedge \neg r_3)$

Seja $\phi_1 = \neg p_1 \wedge p_2 \wedge p_3$ e $\phi_2 = \neg q_1 \wedge q_2 \wedge \neg q_3$.

Renomeamento

- 1 Escolhemos um conjunto de subfórmulas $R \subseteq SF(\phi)$.
- 2 Para cada $\psi \in R$:
 - 1 Escolhemos um símbolo proposicional novo $s(\psi) \in \mathcal{P}$.
 - 2 Trocamos todas as ocorrências de ψ por $s(\psi)$.
 - 3 Incluímos a definição $s(\psi) \rightarrow \psi$ em conjunção.

Exemplo: $(\neg p_1 \wedge p_2 \wedge p_3) \vee (\neg q_1 \wedge q_2 \wedge \neg q_3) \vee (r_1 \wedge r_2 \wedge \neg r_3)$

Seja $\phi_1 = \neg p_1 \wedge p_2 \wedge p_3$ e $\phi_2 = \neg q_1 \wedge q_2 \wedge \neg q_3$.

Escolhendo $R = \{\phi_1, \phi_2\}$, $s(\phi_1) = a$ e $s(\phi_2) = b$, temos

Renomeamento

- 1 Escolhemos um conjunto de subfórmulas $R \subseteq SF(\phi)$.
- 2 Para cada $\psi \in R$:
 - 1 Escolhemos um símbolo proposicional novo $s(\psi) \in \mathcal{P}$.
 - 2 Trocamos todas as ocorrências de ψ por $s(\psi)$.
 - 3 Incluímos a definição $s(\psi) \rightarrow \psi$ em conjunção.

Exemplo: $(\neg p_1 \wedge p_2 \wedge p_3) \vee (\neg q_1 \wedge q_2 \wedge \neg q_3) \vee (r_1 \wedge r_2 \wedge \neg r_3)$

Seja $\phi_1 = \neg p_1 \wedge p_2 \wedge p_3$ e $\phi_2 = \neg q_1 \wedge q_2 \wedge \neg q_3$.

Escolhendo $R = \{\phi_1, \phi_2\}$, $s(\phi_1) = a$ e $s(\phi_2) = b$, temos

$$(a \vee b \vee (r_1 \wedge r_2 \wedge \neg r_3)) \wedge (a \rightarrow (\neg p_1 \wedge p_2 \wedge p_3)) \wedge (b \rightarrow (\neg q_1 \wedge q_2 \wedge \neg q_3))$$

Renomeamento

- 1 Escolhemos um conjunto de subfórmulas $R \subseteq SF(\phi)$.
- 2 Para cada $\psi \in R$:
 - 1 Escolhemos um símbolo proposicional novo $s(\psi) \in \mathcal{P}$.
 - 2 Trocamos todas as ocorrências de ψ por $s(\psi)$.
 - 3 Incluímos a definição $s(\psi) \rightarrow \psi$ em conjunção.

Exemplo: $(\neg p_1 \wedge p_2 \wedge p_3) \vee (\neg q_1 \wedge q_2 \wedge \neg q_3) \vee (r_1 \wedge r_2 \wedge \neg r_3)$

Seja $\phi_1 = \neg p_1 \wedge p_2 \wedge p_3$ e $\phi_2 = \neg q_1 \wedge q_2 \wedge \neg q_3$.

Escolhendo $R = \{\phi_1, \phi_2\}$, $s(\phi_1) = a$ e $s(\phi_2) = b$, temos

$$(a \vee b \vee (r_1 \wedge r_2 \wedge \neg r_3)) \wedge (a \rightarrow (\neg p_1 \wedge p_2 \wedge p_3)) \wedge (b \rightarrow (\neg q_1 \wedge q_2 \wedge \neg q_3))$$

Não preserva equivalência.

Renomeamento

- 1 Escolhemos um conjunto de subfórmulas $R \subseteq SF(\phi)$.
- 2 Para cada $\psi \in R$:
 - 1 Escolhemos um símbolo proposicional novo $s(\psi) \in \mathcal{P}$.
 - 2 Trocamos todas as ocorrências de ψ por $s(\psi)$.
 - 3 Incluímos a definição $s(\psi) \rightarrow \psi$ em conjunção.

Exemplo: $(\neg p_1 \wedge p_2 \wedge p_3) \vee (\neg q_1 \wedge q_2 \wedge \neg q_3) \vee (r_1 \wedge r_2 \wedge \neg r_3)$

Seja $\phi_1 = \neg p_1 \wedge p_2 \wedge p_3$ e $\phi_2 = \neg q_1 \wedge q_2 \wedge \neg q_3$.

Escolhendo $R = \{\phi_1, \phi_2\}$, $s(\phi_1) = a$ e $s(\phi_2) = b$, temos

$$(a \vee b \vee (r_1 \wedge r_2 \wedge \neg r_3)) \wedge (a \rightarrow (\neg p_1 \wedge p_2 \wedge p_3)) \wedge (b \rightarrow (\neg q_1 \wedge q_2 \wedge \neg q_3))$$

Não preserva equivalência. Mas preserva satisfatibilidade!

Reduzindo o número de cláusulas

Contando cláusulas

Denotamos o *número de cláusulas* geradas por ϕ ao ser colocada na FNC por $p(\phi)$.

Reduzindo o número de cláusulas

Contando cláusulas

Denotamos o *número de cláusulas* geradas por ϕ ao ser colocada na FNC por $p(\phi)$.

Forma de ϕ	$p(\phi)$
$\phi_1 \wedge \dots \wedge \phi_n$	$p(\phi_1) + \dots + p(\phi_n)$
$\phi_1 \vee \dots \vee \phi_n$	$p(\phi_1) \cdot \dots \cdot p(\phi_n)$
x ou $\neg x, x \in \mathcal{P}$	1

Reduzindo o número de cláusulas

Contando cláusulas

Denotamos o *número de cláusulas* geradas por ϕ ao ser colocada na FNC por $p(\phi)$.

Forma de ϕ	$p(\phi)$
$\phi_1 \wedge \dots \wedge \phi_n$	$p(\phi_1) + \dots + p(\phi_n)$
$\phi_1 \vee \dots \vee \phi_n$	$p(\phi_1) \cdot \dots \cdot p(\phi_n)$
x ou $\neg x, x \in \mathcal{P}$	1

Exemplo: $\phi = (\neg p_1 \wedge p_2 \wedge p_3) \vee (\neg q_1 \wedge q_2 \wedge \neg q_3) \vee (r_1 \wedge r_2 \wedge \neg r_3)$

Reduzindo o número de cláusulas

Contando cláusulas

Denotamos o *número de cláusulas* geradas por ϕ ao ser colocada na FNC por $p(\phi)$.

Forma de ϕ	$p(\phi)$
$\phi_1 \wedge \dots \wedge \phi_n$	$p(\phi_1) + \dots + p(\phi_n)$
$\phi_1 \vee \dots \vee \phi_n$	$p(\phi_1) \cdot \dots \cdot p(\phi_n)$
x ou $\neg x, x \in \mathcal{P}$	1

Exemplo: $\phi = (\neg p_1 \wedge p_2 \wedge p_3) \vee (\neg q_1 \wedge q_2 \wedge \neg q_3) \vee (r_1 \wedge r_2 \wedge \neg r_3)$

Temos que

$$p(\phi) = (1 + 1 + 1)(1 + 1 + 1)(1 + 1 + 1) = 3^3 = 27$$

Reduzindo o número de cláusulas

O problema

Problema

Escolher $R \subseteq SF(\phi)$ de modo que o número de cláusulas $p(\phi, R)$ da transformação por renomeamento seja mínimo.

Reduzindo o número de cláusulas

Algoritmo de Boy de la Tour

Árvores lineares

Seja ϕ uma fórmula na FNN. Se cada subfórmula de ϕ ocorre somente uma vez, dizemos que ϕ é uma *árvore linear*.

Reduzindo o número de cláusulas

Algoritmo de Boy de la Tour

Árvores lineares

Seja ϕ uma fórmula na FNN. Se cada subfórmula de ϕ ocorre somente uma vez, dizemos que ϕ é uma *árvore linear*.

Se ϕ é uma árvore linear, o algoritmo de Boy de la Tour encontra um conjunto $R \subseteq SF(\phi)$ tal que $p(\phi, R)$ é ótimo (mínimo).

Reduzindo o número de cláusulas

Algoritmo de Boy de la Tour

Árvores lineares

Seja ϕ uma fórmula na FNN. Se cada subfórmula de ϕ ocorre somente uma vez, dizemos que ϕ é uma *árvore linear*.

Se ϕ é uma árvore linear, o algoritmo de Boy de la Tour encontra um conjunto $R \subseteq SF(\phi)$ tal que $p(\phi, R)$ é ótimo (mínimo).

Seu custo de tempo no pior caso é $O(|SF(\phi)|^2)$.

Reduzindo o número de cláusulas

Algoritmo de Boy de la Tour

É feita uma busca na fórmula, incluindo cada $\psi \sqsubset \phi$ que satisfaz

$$a_{\psi}^{\phi} \cdot p(\psi) > a_{\psi}^{\phi} + p(\psi)$$

Reduzindo o número de cláusulas

Algoritmo de Boy de la Tour

É feita uma busca na fórmula, incluindo cada $\psi \sqsubset \phi$ que satisfaz

$$a_{\psi}^{\phi} \cdot p(\psi) > a_{\psi}^{\phi} + p(\psi)$$

Exemplo:

TODO

Conteúdo

- 1 Introdução
- 2 Referencial teórico
- 3 O algoritmo**
- 4 Resultados experimentais
- 5 Conclusão
- 6 Referências

Uma afirmação

Afirmação

Seja $R \subseteq SF(\phi)$ um renomeamento ótimo entre os que contêm no máximo j subfórmulas. Então $R - \{\psi\}$ é ótimo entre os que não consideram ψ e contêm no máximo $j - 1$ subfórmulas.

Uma afirmação

Afirmação

Seja $R \subseteq SF(\phi)$ um renomeamento ótimo entre os que contêm no máximo j subfórmulas. Então $R - \{\psi\}$ é ótimo entre os que não consideram ψ e contêm no máximo $j - 1$ subfórmulas.

Contraexemplo:
TODO

Uma afirmação

Logo, a afirmação não é verdadeira.

Uma afirmação

Logo, a afirmação não é verdadeira.
Mas a usaremos como heurística!

Uma fórmula recursiva

Seja $SF(\phi) = \{\phi_1, \dots, \phi_n\}$ e denote por $f(i, j)$ um renomeamento ótimo entre os que contêm no máximo j subfórmulas e consideram somente as subfórmulas em $\{\phi_1, \dots, \phi_i\}$.

Uma fórmula recursiva

Seja $SF(\phi) = \{\phi_1, \dots, \phi_n\}$ e denote por $f(i, j)$ um renomeamento ótimo entre os que contêm no máximo j subfórmulas e consideram somente as subfórmulas em $\{\phi_1, \dots, \phi_i\}$. Então

$$f(i, 0) = f(0, j) = \emptyset, \forall i, j$$

Uma fórmula recursiva

Seja $SF(\phi) = \{\phi_1, \dots, \phi_n\}$ e denote por $f(i, j)$ um renomeamento ótimo entre os que contêm no máximo j subfórmulas e consideram somente as subfórmulas em $\{\phi_1, \dots, \phi_i\}$. Então

$$f(i, 0) = f(0, j) = \emptyset, \forall i, j$$

e

$$f(i, j) = \begin{cases} f(i-1, j-1) \cup \{\phi_i\} & \text{se } p(\phi, f(i-1, j-1) \cup \{\phi_i\}) < p(\phi, f(i-1, j)) \\ f(i-1, j) & \text{caso contrário} \end{cases}$$

Uma fórmula recursiva

Seja $SF(\phi) = \{\phi_1, \dots, \phi_n\}$ e denote por $f(i, j)$ um renomeamento ótimo entre os que contêm no máximo j subfórmulas e consideram somente as subfórmulas em $\{\phi_1, \dots, \phi_i\}$. Então

$$f(i, 0) = f(0, j) = \emptyset, \forall i, j$$

e

$$f(i, j) = \begin{cases} f(i-1, j-1) \cup \{\phi_i\} & \text{se } p(\phi, f(i-1, j-1) \cup \{\phi_i\}) < p(\phi, f(i-1, j)) \\ f(i-1, j) & \text{caso contrário} \end{cases}$$

Queremos $f(n, n)$!

Uma implementação por computação ascendente

- 1: seja $dp[0..n]$ um novo arranjo com $dp[j] = \emptyset$ para todo j
- 2: **para** $i \leftarrow 1$ **até** n **faça**
- 3: **para** $j \leftarrow n$ **descendo até** 1 **faça**
- 4: $alt \leftarrow dp[j - 1] \cup \{\phi_i\}$
- 5: **se** $p(\phi, alt) < p(\phi, dp[j])$ **então**
- 6: $dp[j] \leftarrow alt$
- 7: **fim se**
- 8: **fim para**
- 9: **fim para**

Uma implementação por computação ascendente

- 1: seja $dp[0..n]$ um novo arranjo com $dp[j] = \emptyset$ para todo j
- 2: **para** $i \leftarrow 1$ **até** n **faça**
- 3: **para** $j \leftarrow n$ **descendo até** 1 **faça**
- 4: $alt \leftarrow dp[j - 1] \cup \{\phi_i\}$
- 5: **se** $p(\phi, alt) < p(\phi, dp[j])$ **então**
- 6: $dp[j] \leftarrow alt$
- 7: **fim se**
- 8: **fim para**
- 9: **fim para**

O custo de tempo no pior caso é $O(|SF(\phi)|^3)$.

Conjectura para árvores lineares

Conjectura

Se ϕ é uma árvore linear e $SF(\phi) = \{\phi_1, \dots, \phi_n\}$, então $f(n, n)$ é ótimo.

Conjectura para árvores lineares

Conjectura

Se ϕ é uma árvore linear e $SF(\phi) = \{\phi_1, \dots, \phi_n\}$, então $f(n, n)$ é ótimo.

Apresentamos resultados experimentais para a conjectura na próxima seção.

Conteúdo

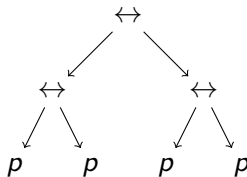
- 1 Introdução
- 2 Referencial teórico
- 3 O algoritmo
- 4 Resultados experimentais**
- 5 Conclusão
- 6 Referências

Metodologia

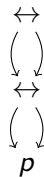
Representações de fórmulas

$$(p \leftrightarrow p) \leftrightarrow (p \leftrightarrow p)$$

Cadeia



Árvore sintática



DAG

Metodologia

Implementação

Foi implementado um programa em C++ 11 que realiza, em ordem, as seguintes transformações:

Metodologia

Implementação

Foi implementado um programa em C++ 11 que realiza, em ordem, as seguintes transformações:

- 1 Análise sintática

Metodologia

Implementação

Foi implementado um programa em C++ 11 que realiza, em ordem, as seguintes transformações:

- 1 Análise sintática
- 2 Conversão para FNN

Metodologia

Implementação

Foi implementado um programa em C++ 11 que realiza, em ordem, as seguintes transformações:

- 1 Análise sintática
- 2 Conversão para FNN
- 3 Aplainamento

Metodologia

Implementação

Foi implementado um programa em C++ 11 que realiza, em ordem, as seguintes transformações:

- 1 Análise sintática
- 2 Conversão para FNN
- 3 Aplainamento
- 4 Conversão para DAG

Metodologia

Implementação

Foi implementado um programa em C++ 11 que realiza, em ordem, as seguintes transformações:

- 1 Análise sintática
- 2 Conversão para FNN
- 3 Aplainamento
- 4 Conversão para DAG
- 5 Renomeamento

Metodologia

Implementação

Foi implementado um programa em C++ 11 que realiza, em ordem, as seguintes transformações:

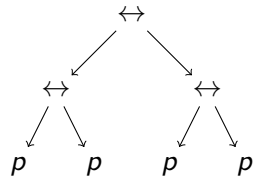
- 1 Análise sintática
- 2 Conversão para FNN
- 3 Aplainamento
- 4 Conversão para DAG
- 5 Renomeamento
- 6 Conversão para FNC

Metodologia

Implementação – Análise sintática

$(p \leftrightarrow p) \leftrightarrow (p \leftrightarrow p)$

\mapsto



Cadeia

Árvore sintática

Metodologia

Implementação – Conversão para FNN

Coloca-se a fórmula na forma normal negada.

Metodologia

Implementação – Conversão para FNN

Coloca-se a fórmula na forma normal negada.

- 1 Simplifica a implementação dos algoritmos de renomeamento e a conversão para FNC.

Metodologia

Implementação – Conversão para FNN

Coloca-se a fórmula na forma normal negada.

- 1 Simplifica a implementação dos algoritmos de renomeamento e a conversão para FNC.
- 2 Permite testar nossa conjectura para árvores lineares.

Metodologia

Implementação – Aplainamento

$$p \wedge (q \wedge r) \mapsto p \wedge q \wedge r$$

e

$$p \vee (q \vee r) \mapsto p \vee q \vee r$$

Metodologia

Implementação – Aplainamento

$$p \wedge (q \wedge r) \mapsto p \wedge q \wedge r$$

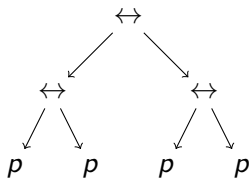
e

$$p \vee (q \vee r) \mapsto p \vee q \vee r$$

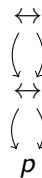
Viabiliza mais simplificações!

Metodologia

Implementação – Conversão para DAG



Árvore sintática



DAG

Metodologia

Experimentos propostos

Conteúdo

- 1 Introdução
- 2 Referencial teórico
- 3 O algoritmo
- 4 Resultados experimentais
- 5 Conclusão**
- 6 Referências

content...

Conteúdo

- 1 Introdução
- 2 Referencial teórico
- 3 O algoritmo
- 4 Resultados experimentais
- 5 Conclusão
- 6 Referências**

Referências I

- [1] R. Bloem, U. Egly, P. Klampfl, R. Könighofer, and F. Lonsing, “SAT-based methods for circuit synthesis,” in *Proceedings of the 14th Conference on Formal Methods in Computer-Aided Design*, pp. 31–34, FMCAD Inc, 2014.
- [2] R. Nieuwenhuis and A. Oliveras, “On SAT modulo theories and optimization problems,” in *Theory and Applications of Satisfiability Testing-SAT 2006*, pp. 156–169, Springer, 2006.
- [3] A. Gupta, M. K. Ganai, and C. Wang, “SAT-based verification methods and applications in hardware verification,” in *Formal Methods for Hardware Verification*, pp. 108–143, Springer, 2006.

Referências II

- [4] J. Harrison, *Handbook of practical logic and automated reasoning*.
Cambridge University Press, 2009.
- [5] E. J. Horvitz, *Automated reasoning for biology and medicine*.
Knowledge Systems Laboratory, Section on Medical
Informatics, Stanford University, 1992.
- [6] S. A. Cook, “The complexity of theorem-proving procedures,”
in *Proceedings of the third annual ACM symposium on
Theory of computing*, pp. 151–158, ACM, 1971.

Referências III

- [7] M. Davis and H. Putnam, “A computing procedure for quantification theory,” *Journal of the ACM (JACM)*, vol. 7, no. 3, pp. 201–215, 1960.
- [8] M. Davis, G. Logemann, and D. Loveland, “A machine program for theorem-proving,” *Communications of the ACM*, vol. 5, no. 7, pp. 394–397, 1962.
- [9] A. Biere, M. Heule, H. van Maaren, and T. Walsh, “Conflict-driven clause learning SAT solvers,” *Handbook of Satisfiability, Frontiers in Artificial Intelligence and Applications*, pp. 131–153, 2009.

Referências IV

- [10] T. Boy de la Tour, “An optimality result for clause form translation,” *Journal of Symbolic Computation*, vol. 14, no. 4, pp. 283–301, 1992.
- [11] P. Jackson and D. Sheridan, “Clause form conversions for boolean circuits,” in *Theory and applications of satisfiability testing*, pp. 183–198, Springer, 2004.