# Lógica para Computação

## DPLL

Thiago Alves Rocha

*thiagoalvesifce@gmail.com*

# Tópicos

# Introdução

- A ideia é construir uma valoração para uma fórmula fornecida em CNF
- A cada passo do algoritmo, um literal $L$ é escolhido
- Primeiro, fazemos $v(L) = T$

# Introdução

- Se a fórmula em CNF for $\{\{a, \neg d\}, \{\neg b, \neg c\}, \{c, a\}\}$ e fizermos $v(a) = T$?

# Introdução

- Se a fórmula em CNF for $\{\{a, \neg d\}, \{\neg b, \neg c\}, \{c, a\}\}$ e fizermos $v(a) = T$?
- $\{\{\neg b, \neg c\}\}$
- Basta um literal na cláusula ser verdadeiro

# Introdução

- Se a fórmula em CNF for $\{\{a, \neg d\}, \{\neg b, \neg c\}, \{c, a\}\}$ e fizermos $v(a) = T$?
- $\{\{\neg b, \neg c\}\}$
- Basta um literal na cláusula ser verdadeiro
- Já podemos ver que uma das vantagens do DPLL é não precisar atribuir um valor verdade para todas as atômicas
- A atômica $d$ não precisou ser avaliada quando fizemos $v(a) = T$

# Introdução

- Se a fórmula em CNF for $\{\{a, \neg d\}, \{\neg b, \neg c\}, \{c, \neg a\}\}$ e fizermos $v(a) = T$?

# Introdução

- Se a fórmula em CNF for $\{\{a, \neg d\}, \{\neg b, \neg c\}, \{c, \neg a\}\}$ e fizermos $v(a) = T$?
- $\{\{\neg b, \neg c\}, \{c\}\}$
- Se $v(a) = T$ então $v(\neg a) = F$
- A cláusula $\{c, \neg a\}$ agora depende apenas de $c$

# DPLL

- A cada iteração do algoritmo, um literal $L$ é escolhido
- Primeira fazemos $v(L) = T$
- Se a simplificação obter a fórmula em CNF $\{\}$?

# DPLL

- A cada iteração do algoritmo, um literal $L$ é escolhido
- Primeira fazemos $v(L) = T$
- Se a simplificação obter a fórmula em CNF $\{\}$?
- Temos que a valoração satisfaz a fórmula

# DPLL

- A cada iteração do algoritmo, um literal $L$ é escolhido
- Primeira fazemos $v(L) = T$
- Se a simplificação obter alguma cláusula vazia?

# DPLL

- A cada iteração do algoritmo, um literal $L$ é escolhido
- Primeira fazemos $v(L) = T$
- Se a simplificação obter alguma cláusula vazia?
- Por exemplo, $\{\{\neg b, \neg c\}, \{\}\}$

# DPLL

- A cada iteração do algoritmo, um literal $L$ é escolhido
- Primeira fazemos $v(L) = T$
- Se a simplificação obter alguma cláusula vazia?
- Por exemplo, $\{\{\neg b, \neg c\}, \{\}\}$
- Temos que a valoração atual **não** satisfaz a fórmula

# DPLL

- A cada iteração do algoritmo, um literal $L$ é escolhido
- Primeira fazemos $v(L) = T$
- Se a simplificação obter alguma cláusula vazia?
- Por exemplo, $\{\{\neg b, \neg c\}, \{\}\}$
- Temos que a valoração atual **não** satisfaz a fórmula
- Ainda é possível que a fórmula seja satisfatível?

# DPLL

- A cada iteração do algoritmo, um literal $L$ é escolhido
- Primeira fazemos $v(L) = T$
- Se a simplificação obter alguma cláusula vazia?
- Por exemplo, $\{\{\neg b, \neg c\}, \{\}\}$
- Temos que a valoração atual **não** satisfaz a fórmula
- Ainda é possível que a fórmula seja satisfatível?
- Fazer $v(L) = F$

# DPLL

- A cada iteração do algoritmo, um literal $L$ é escolhido
- Primeira fazemos $v(L) = T$
- Se a simplificação não obter a fórmula vazia e nenhuma cláusula vazia?

# DPLL

- A cada iteração do algoritmo, um literal $L$ é escolhido
- Primeira fazemos $v(L) = T$
- Se a simplificação não obter a fórmula vazia e nenhuma cláusula vazia?
- Por exemplo, $\{\{\neg b, \neg c\}\}$?

- A cada iteração do algoritmo, um literal $L$ é escolhido
- Primeira fazemos $v(L) = T$
- Se a simplificação não obter a fórmula vazia e nenhuma cláusula vazia?
- Por exemplo, $\{\{\neg b, \neg c\}\}$?
- Outro literal deve ser escolhido e o processo continua até achar uma valoração para satisfazer a fórmula ou não tiver mais atômicas para serem testadas

# Propagação de Unidade

- Seja a fórmula em CNF $S = \{\{r\}, \{p, q, \neg r\}, \{r, p, \neg q\}, \{\neg q, p\}\}$.
- Para $\varphi$ ser satisfatível, a cláusula unitária $\{r\}$ deve ser ser verdadeira, ou seja, $r$ deve ser verdadeiro.
- Com isso, para $\varphi$ ser satisfatível, $\{\{p, q\}, \{\neg q, p\}\}$ deve ser satisfatível.

**Teorema**

Seja $\{l\} \in S$ uma cláusula unitária e seja $S'$ obtido de $S$ deletando toda cláusula com $l$ e removendo $\neg l$ das clausulas restantes. Então $S$ é satisfatível se e somente se $S'$ é satisfatível.

# Propagação de Unidade

## Prova

$\Rightarrow$ Suponha $S$ satisfatível. Logo, existe uma valoração $v$ tal que $v(S) = T$.
Como $\{l\} \in S$, $v(l) = T$ e $v(\neg l) = F$.
Seja $v'$ uma valoração que concorda com $v$ em todas as atômicas de $S'$.
Seja $C'$ uma cláusula qualquer de $S'$. Se $C' \in S$, então $v'(C) = T$.
Se $C' \notin S$, ou seja, $C' = C - \{\neg l\}$ em que $C$ é uma cláusula de $S$.
Como $v(\neg l) = F$ e $v(C) = T$ então existe algum literal $l_2 \in C$ tal que $v(l_2) = T$. Dessa forma, $l_2 \in C'$, ou seja, $v'(C') = T$.
$\Leftarrow$ Suponha que $S'$ é satisfatível. Portanto, existe $v'$ tal que $v'(S') = T$.
Seja $C' \in S'$ uma cláusula, ou seja, $v'(C') = T$.
Como $\{l\} \in S$, seja $v = v' \cup \{(l, T)\}$, ou seja, $v(l) = T$ e $v(\neg l) = F$.
Seja $C$ uma cláusula qualquer de $S$. Se $C \in S'$, $v(C) = T$.
Se $C \notin S'$, então $l \in C$ ou $\neg l \in C$. Se $l \in C$, então $v(C) = T$.
Se $\neg l \in C$, então $C - \{\neg l\} \in S'$ e $v'(C - \{\neg l\}) = T$. Dessa forma, $v(C) = T$.

# Regra da Divisão

**Teorema**

Seja $\alpha$ uma fórmula em CNF, e seja $l$ um literal que ocorre em $\alpha$. Então $\alpha$ é satisfatível se e somente se $\alpha \cup \{l\}$ é satisfatível ou $\alpha \cup \{\neg l\}$ é satisfatível.

# Algoritmo DPLL

```
function DPLLCheck(φ)
    clauses ← CNF(φ)
    return DPLL(clauses, {})
```

```
function DPLL(clauses, val)
    clauses, val ← SIMPLIFICA(clauses, val)
    if clauses == {} then
        return val
    if {} ∈ clauses then
        return False
    atomic ← first(atoms(clauses))
    result ← DPLL(clauses ∪ {atomic}, val)
    if result ≠ False then
        return result
    return DPLL(clauses ∪ {¬atomic}, val)
```

# Algoritmo DPLL

```
function DPLLCheck(φ)
    clauses ← CNF(φ)
    return DPLL(clauses, {})
```

```
function DPLL(clauses, val)
    clauses, val ← SIMPLIFICA(clauses, val)
    if clauses == {} then
        return val
    if {} ∈ clauses then
        return False
    atomic ← first(atoms(clauses))
    result ← DPLL(clauses ∪ {atomic}, val)
    if result ≠ False then
        return result
    return DPLL(clauses ∪ {¬atomic}, val)
```

- Heurísticas podem ser utilizadas na escolha da atômica

# Simplificação

- Note que o algoritmo **DPLL** faz uma chamada da função **SIMPLIFICA**
- Note também que fazemos uma chamada recursiva de **DPLL**$(clauses \cup \{atomic\}, val)$
- Na chamada recursiva, o algoritmo vai chamar a função de simplificação
- A função de simplificação definida a seguir vai eliminar cláusulas unitárias como $\{L\}$ em que $L$ é literal
- Também vai eliminar o complemento de $L$ das outras cláusulas
- O raciocínio é análogo para **DPLL**$(clauses \cup \{\neg atomic\})$

# Algoritmo Simplifica

**function** SIMPLIFICA($clauses, val$)
  **while** temUnitaria(clauses) **do**
    $literal \leftarrow literalUnitaria(clauses)$
    $val \leftarrow val \cup \{literal\}$
    $clauses \leftarrow removeClausesWith(clauses, literal)$
    $clauses \leftarrow removeComplementLit(clauses, literal)$
  **return** $clauses, val$

# Simplificação

- No algoritmo **Simplifica**, depois de retirar o complemento de $L$ das cláusulas restantes é possível que apareça novas cláusulas unitárias
- Outras simplificações podem ser feitas
- O **while** do algoritmo será executado até não existir mais cláusulas unitárias
- A seguir vamos ver um exemplo de execução do algoritmo **Simplifica**
- E em seguida vamos analisar exemplos de execução do algoritmo **DPLL**

# Exemplo

## Exemplo

Vamos executar
**SIMPLIFICA**$(\{\{\neg x_1\}, \{x_1, x_2\}, \{x_1, x_3\}, \{\neg x_2, \neg x_3, \neg x_4, \neg x_5\}, \{\neg x_1, x_4\}\})$

# Exemplo

**Exemplo**

Vamos executar **DPLL**$(\{\{\neg a, b\}, \{b, c\}, \{a, \neg b, \neg c\}, \{a, \neg b, c\}\})$

# Exemplo

## Exemplo

Vamos executar $\textbf{DPLL}(\{\{a, b\}, \{\neg a, b\}, \{a, \neg b\}, \{\neg a, \neg b\}\})$