ALGORITMOS PARA O PROBLEMA DA SATISFAZIBILIDADE

INSTITUTO
DE INFORMÁTICA
UFRGS

Isis Burmeister Pericolo
Leonardo Holtz de Oliveira
Milena Lopes Maciel
Dezembro de 2017 - Porto Alegre, RS

FÓRMULAS PROPOSICIONAIS

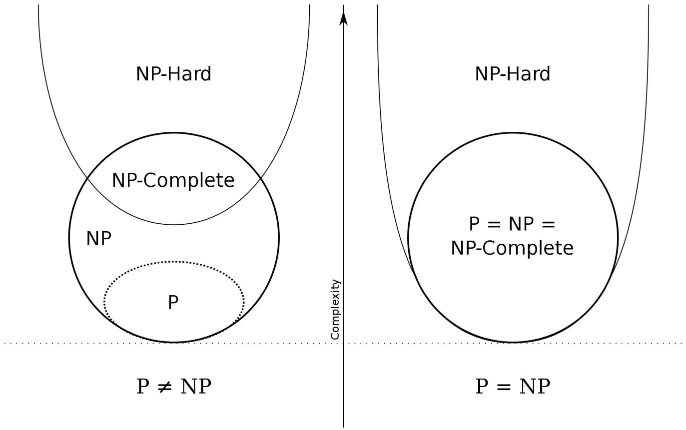
$$
\begin{aligned}
\neg(A \wedge B) &= \neg A \vee \neg B \\
\neg(A \vee B) &= \neg A \wedge \neg B \\
\neg(\neg A) &= A \\
\neg(A \Rightarrow B) &= A \wedge \neg B \\
A \Rightarrow B &= \neg A \vee B \\
A \Leftrightarrow B &= (A \wedge B) \vee (\neg A \wedge \neg B) \\
\neg(A \Leftrightarrow B) &= (A \wedge \neg B) \vee (\neg A \wedge B)
\end{aligned}
$$

ALGORITMOS

```
// Input arg:        Current decision level d
// Output arg:       Backtrack decision level β
// Return value:     SATISFIABLE or UNSATISFIABLE
//
SAT (d, &β)
{
    if (Decide (d) != DECISION)
        return SATISFIABLE;
    while (TRUE) {
        if (Deduce (d) != CONFLICT) {
            if (SAT (d + 1, β) == SATISFIABLE)
                return SATISFIABLE;
            else if (β != d || d == 0) {
                Erase (d); return UNSATISFIABLE;
            }
        }
        if (Diagnose (d, β) == CONFLICT) {
            return UNSATISFIABLE;
        }
    }
}
```

SAT

COMPLEXIDADE



Tendo em vista que algoritmos NP-completos são difíceis e demorados de resolver, precisamos desenvolver outros métodos para achar e verificar respostas para esses problemas. Aqui, trabalhamos com diversas restrições:

- **FORMA NORMAL DISJUNTIVA**:

Tendo em vista que algoritmos NP-completos são difíceis e demorados de resolver, precisamos desenvolver outros métodos para achar e verificar respostas para esses problemas. Aqui, trabalhamos com diversas <u>restrições</u>:

➔ **FORMA NORMAL DISJUNTIVA**:

$$(x \land y \land z) \lor (a \land \neg b) \lor \ldots \lor (m \land \neg n \land p \land \neg p)$$

Tendo em vista que algoritmos NP-completos são difíceis e demorados de resolver, precisamos desenvolver outros métodos para achar e verificar respostas para esses problemas. Aqui, trabalhamos com diversas restrições:

- **FORMA NORMAL DISJUNTIVA**:

$$(x \wedge y \wedge z) \vee (a \wedge \neg b) \vee \ldots \vee (m \wedge \neg n \wedge p \wedge \neg p)$$

O algoritmo para a satisfazibilidade de uma fórmula assim consiste em simplesmente determinar que um dos implicantes da FND é verdadeiro.

➔ 2-SAT:

➔ **2-SAT**:

- FORMA NORMAL CONJUNTIVA
- ATÉ DOIS LITERAIS EM CADA CLÁUSULA

➔ **2-SAT**:

- FORMA NORMAL CONJUNTIVA
- ATÉ DOIS LITERAIS EM CADA CLÁUSULA

$$(x \lor y) \land (a \land \neg b) \land \ldots \land (m \lor \neg n) \land (p \lor \neg p)$$

➔ 2-SAT:

- FORMA NORMAL CONJUNTIVA
- ATÉ DOIS LITERAIS EM CADA CLÁUSULA

$$(x \vee y) \wedge (a \wedge \neg b) \wedge \ldots \wedge (m \vee \neg n) \wedge (p \vee \neg p)$$

OBS.: também podem ser usados grafos acíclicos direcionados.

➔ 2-SAT:

- FORMA NORMAL CONJUNTIVA
- ATÉ DOIS LITERAIS EM CADA CLÁUSULA

$$(x \lor y) \land (a \land \neg b) \land \ldots \land (m \lor \neg n) \land (p \lor \neg p)$$

OBS.: também podem ser usados grafos acíclicos direcionados.

$$(p \land \neg\neg(\neg q \land \neg\neg p))$$

exemplo de fórmula

➔ 2-SAT:

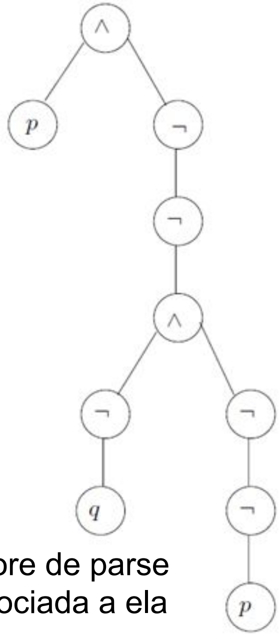
- FORMA NORMAL CONJUNTIVA
- ATÉ DOIS LITERAIS EM CADA CLÁUSULA

$$(x \lor y) \land (a \land \neg b) \land \ldots \land (m \lor \neg n) \land (p \lor \neg p)$$

OBS.: também podem ser usados grafos acíclicos direcionados.

$$(p \land \neg\neg(\neg q \land \neg\neg p))$$

exemplo de fórmula



árvore de parse associada a ela

➔ **2-SAT**:

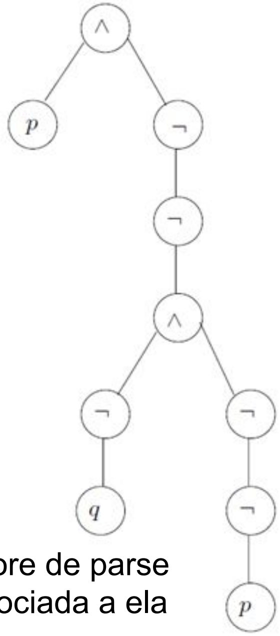
- FORMA NORMAL CONJUNTIVA
- ATÉ DOIS LITERAIS EM CADA CLÁUSULA

$$(x \vee y) \wedge (a \wedge \neg b) \wedge \ldots \wedge (m \vee \neg n) \wedge (p \vee \neg p)$$

OBS.: também podem ser usados grafos acíclicos direcionados.

$$(p \wedge \neg\neg(\neg q \wedge \neg\neg p))$$

exemplo de fórmula



árvore de parse associada a ela

grafo acíclico direcionado da fórmula

➔ **2-SAT**:

- FORMA NORMAL CONJUNTIVA
- ATÉ DOIS LITERAIS EM CADA CLÁUSULA

$$(x \lor y) \land (a \land \neg b) \land \ldots \land (m \lor \neg n) \land (p \lor \neg p)$$

Ainda, podem ser usados como entrada para um 2-SAT, grafos de implicação, que manipulam a FNC através da seguinte equivalência lógica:

$$(x0 \lor \neg x1) \equiv (\neg x0 \Rightarrow \neg x1) \equiv (x1 \Rightarrow x0)$$

➔ **2-SAT**:

Um algoritmo para a resolução do 2-SAT foi descrito por Krom em 1967;

➔ 2-SAT:

A fórmula de entrada tem duas cláusulas com uma mesma variável e esta está negada em uma delas;

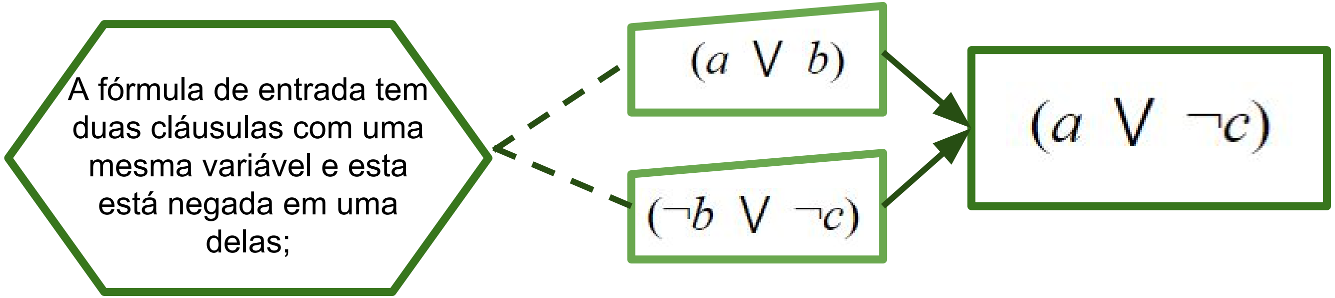
➔ 2-SAT:

A fórmula de entrada tem duas cláusulas com uma mesma variável e esta está negada em uma delas;

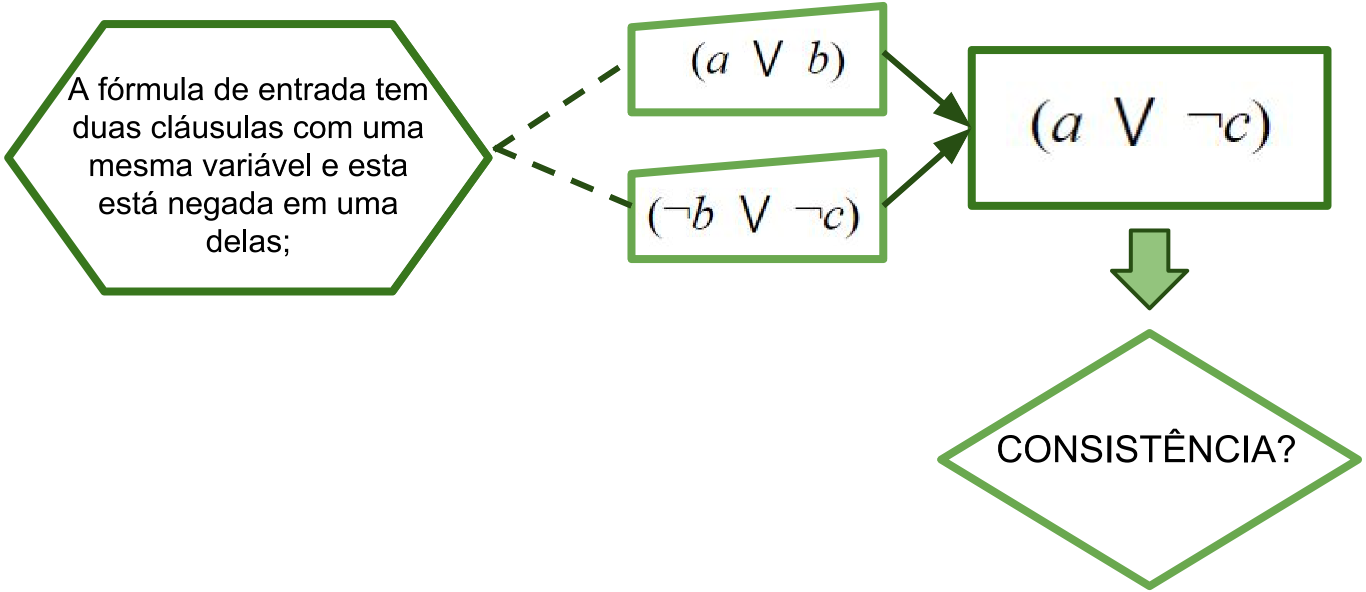
$(a \lor b)$

$(\neg b \lor \neg c)$

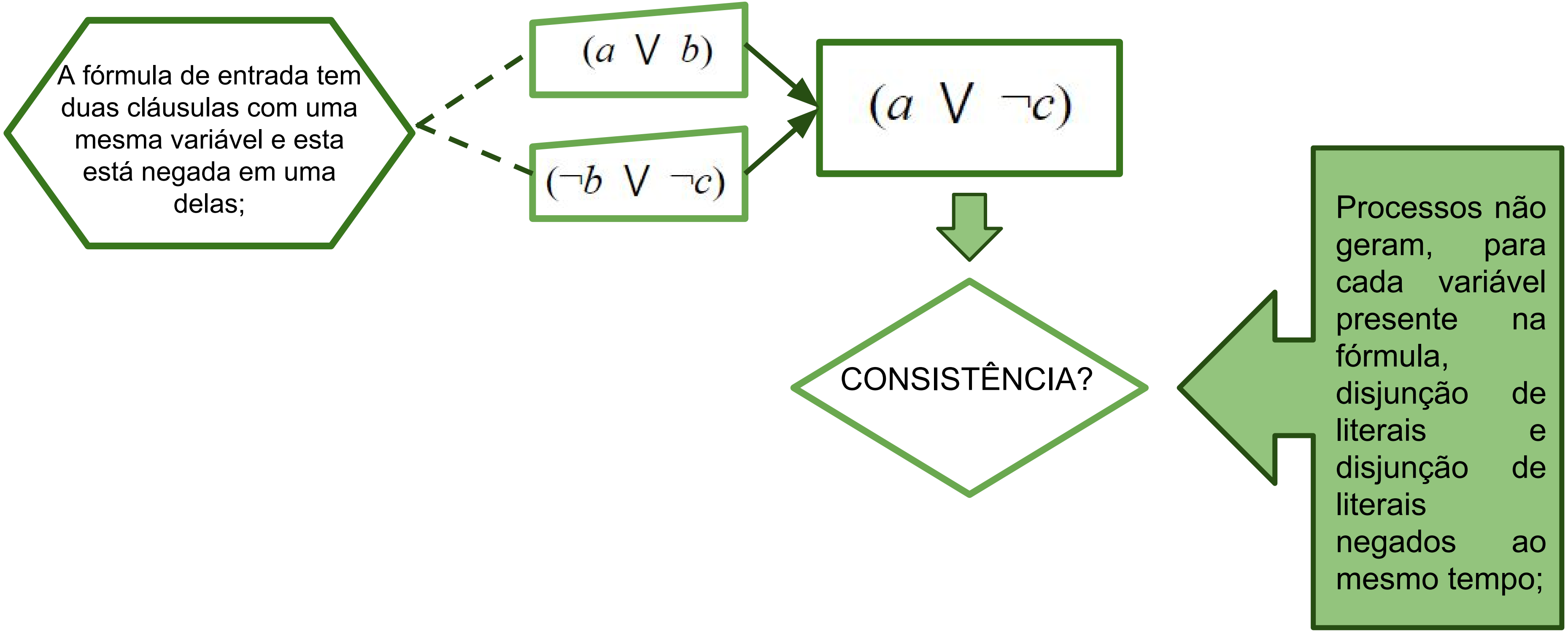
➔ 2-SAT:


A fórmula de entrada tem duas cláusulas com uma mesma variável e esta está negada em uma delas;
$(a \vee b)$
$(\neg b \vee \neg c)$
$(a \vee \neg c)$

➔ 2-SAT:

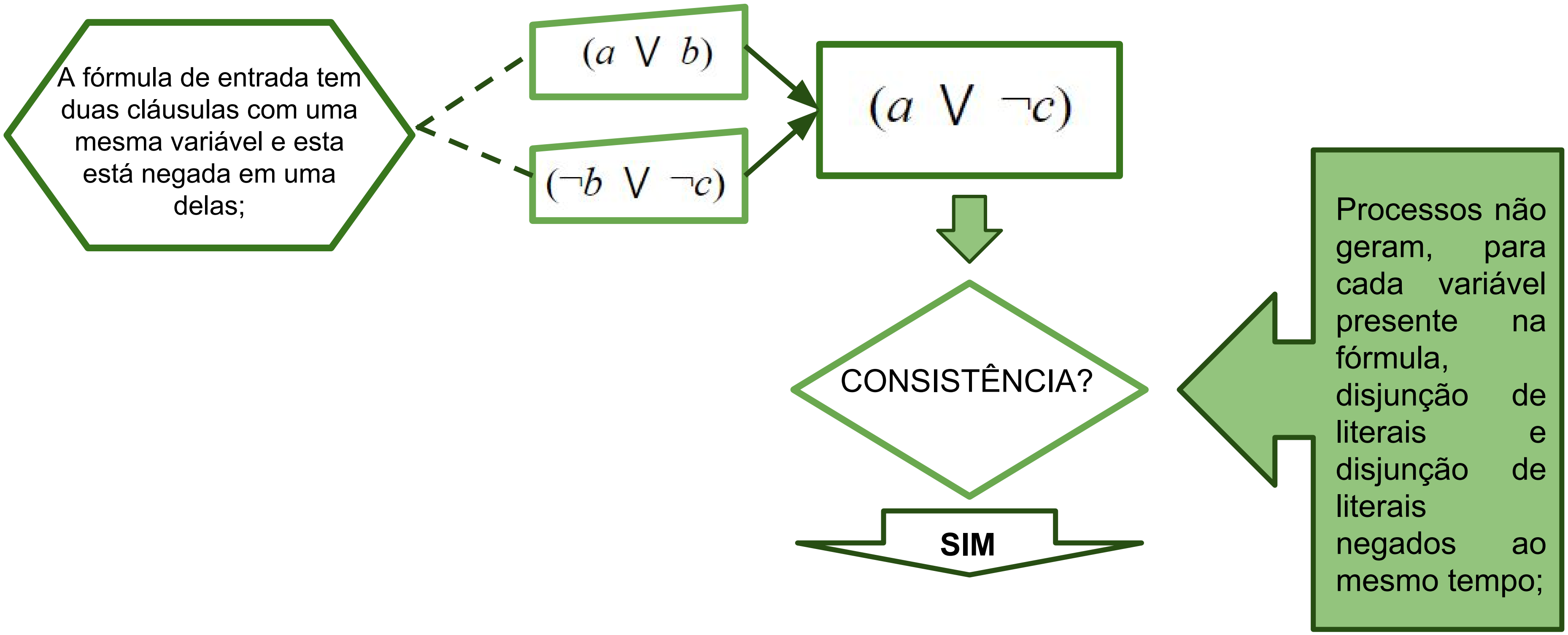

A fórmula de entrada tem duas cláusulas com uma mesma variável e esta está negada em uma delas;
$(a \vee b)$
$(\neg b \vee \neg c)$
$(a \vee \neg c)$
CONSISTÊNCIA?

→ 2-SAT:

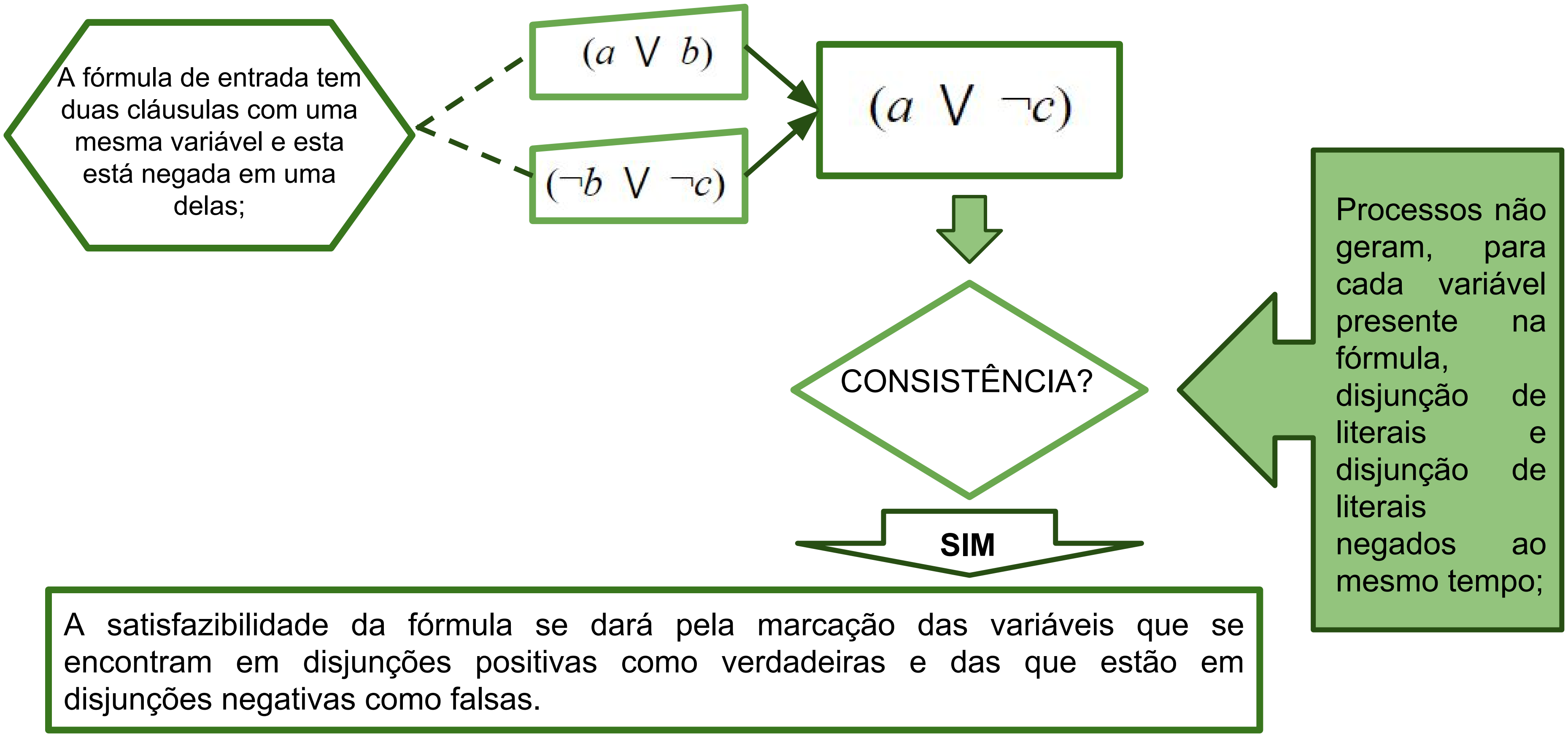


A fórmula de entrada tem duas cláusulas com uma mesma variável e esta está negada em uma delas;
$(a \vee b)$
$(\neg b \vee \neg c)$
$(a \vee \neg c)$
CONSISTÊNCIA?
Processos não geram, para cada variável presente na fórmula, disjunção de literais e disjunção de literais negados ao mesmo tempo;

➔ 2-SAT:


A fórmula de entrada tem duas cláusulas com uma mesma variável e esta está negada em uma delas;
$(a \vee b)$
$(\neg b \vee \neg c)$
$(a \vee \neg c)$
CONSISTÊNCIA?
SIM
Processos não geram, para cada variável presente na fórmula, disjunção de literais e disjunção de literais negados ao mesmo tempo;

➔ 2-SAT:



A fórmula de entrada tem duas cláusulas com uma mesma variável e esta está negada em uma delas;
$(a \lor b)$
$(\neg b \lor \neg c)$
$(a \lor \neg c)$
CONSISTÊNCIA?
Processos não geram, para cada variável presente na fórmula, disjunção de literais e disjunção de literais negados ao mesmo tempo;
SIM
A satisfazibilidade da fórmula se dará pela marcação das variáveis que se encontram em disjunções positivas como verdadeiras e das que estão em disjunções negativas como falsas.

➔ **HORN-SAT**:

- FÓRMULAS DE HORN

➔ **HORN-SAT**:

- FÓRMULAS DE HORN
  - conjunção de cláusulas de Horn

➔ **HORN-SAT**:

- FÓRMULAS DE HORN
    - conjunção de cláusulas de Horn

$$(\neg x1 \lor \neg x2 \lor \ldots \lor \neg xn \lor y)$$

➔ HORN-SAT:

- FÓRMULAS DE HORN
    - conjunção de cláusulas de Horn

$$(\neg x1 \lor \neg x2 \lor \ldots \lor \neg xn \lor y)$$

$$(x1 \land x2 \land \ldots \land xn) \Rightarrow y$$

➔ HORN-SAT:

- FÓRMULAS DE HORN
  - conjunção de cláusulas de Horn

$$(\neg x1 \;\vee \neg x2 \;\vee \ldots \vee \neg xn \;\vee y)$$

O algoritmo de satisfazibilidade usando fórmulas de Horn utiliza da regra de propagação unitária: *se na fórmula existe uma cláusula com um único literal x*, então **todas as cláusulas contendo x** (exceto a unitária de origem) e **todas as cláusulas contendo ¬x são removidas**;

➔ HORN-SAT:

- FÓRMULAS DE HORN
  - conjunção de cláusulas de Horn

$$(\neg x1 \lor \neg x2 \lor \ldots \lor \neg xn \lor y)$$

O algoritmo de satisfazibilidade usando fórmulas de Horn utiliza da regra de propagação unitária: *se na fórmula existe uma cláusula com um único literal x*, então **todas as cláusulas contendo x** (exceto a unitária de origem) e **todas as cláusulas contendo ¬x são removidas**; .

Para acharmos um conjunto de valores para as variáveis que satisfaça a fórmula final, marcamos todas as variáveis que estão em cláusulas unitárias (sozinhas dentro do termo) como verdadeiras e todas as outras como falsas.

➔

FÓRMULA SATISFAZÍVEL

➔ HORN-SAT:

- FÓRMULAS DE HORN
  - conjunção de cláusulas de Horn

$$(\neg x1 \ \lor \neg x2 \ \lor \ldots \lor \neg xn \ \lor y)$$

O algoritmo de satisfazibilidade usando fórmulas de Horn utiliza da regra de <u>propagação unitária</u>: *se na fórmula existe uma cláusula com um único literal x*, então **todas as cláusulas contendo x** (exceto a unitária de origem) e **todas as cláusulas contendo ¬x são removidas**;

Se a propagação unitária e as transformações feitas na fórmula geraram um par de cláusulas que sejam (x) e (¬x), não podemos satisfazer as duas, portanto....

➔ HORN-SAT:

- FÓRMULAS DE HORN
  - conjunção de cláusulas de Horn

$$(\neg x1 \ \vee \neg x2 \ \vee \ldots \vee \neg xn \ \vee y)$$

O algoritmo de satisfazibilidade usando fórmulas de Horn utiliza da regra de propagação unitária: *se na fórmula existe uma cláusula com um único literal x*, então **todas as cláusulas contendo x** (exceto a unitária de origem) e **todas as cláusulas contendo ¬x são removidas**;

Se a propagação unitária e as transformações feitas na fórmula geraram um par de cláusulas que sejam (x) e (¬x), não podemos satisfazer as duas, portanto....

→

FÓRMULA INSATISFAZÍVEL

➔ **3-SAT:**

➔ **3-SAT**:

- FORMA NORMAL CONJUNTIVA
- MÁXIMO DE TRÊS VARIÁVEIS EM CADA CLÁUSULA

➔ 3-SAT:

- FORMA NORMAL CONJUNTIVA
- MÁXIMO DE TRÊS VARIÁVEIS EM CADA CLÁUSULA

$$(x \lor x \lor y) \land (\neg x \lor \neg y \lor \neg y) \land (\neg x \lor y \lor y)$$

➔ **3-SAT**:

- FORMA NORMAL CONJUNTIVA
- MÁXIMO DE TRÊS VARIÁVEIS EM CADA CLÁUSULA

FÓRMULA É SATISFAZÍVEL SE AO MENOS UMA DAS VARIÁVEIS EM CADA CLÁUSULA É VERDADEIRA

Um dos exemplos mais comuns de aplicações do 3-SAT é o de encontrar cliques em um dado grafo.

➔ 3-SAT:

- FORMA NORMAL CONJUNTIVA
- MÁXIMO DE TRÊS VARIÁVEIS EM CADA CLÁUSULA

FÓRMULA É SATISFAZÍVEL SE AO MENOS UMA DAS VARIÁVEIS EM CADA CLÁUSULA É VERDADEIRA

Um dos exemplos mais comuns de aplicações do 3-SAT é o de encontrar cliques em um dado grafo.

O grafo tendo n nodos, onde cada um é um dos literais da fórmula proposicional, podemos descobrir se existe um n-clique neste grafo apenas verificando a satisfazibilidade da fórmula.

➔ 3-SAT:

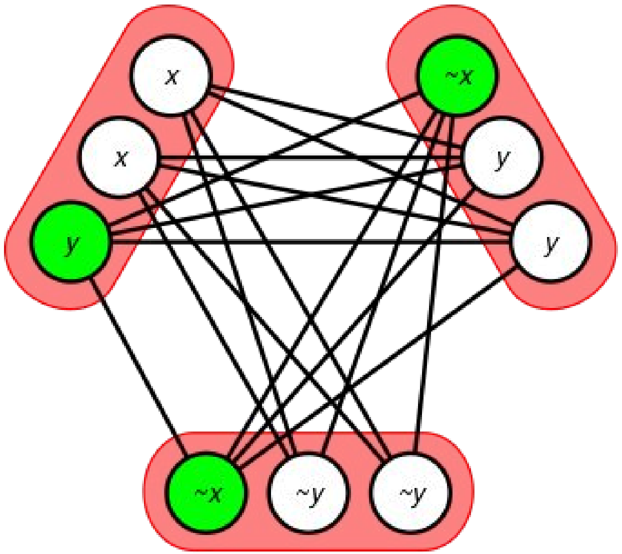
- FORMA NORMAL CONJUNTIVA
- MÁXIMO DE TRÊS VARIÁVEIS EM CADA CLÁUSULA

FÓRMULA É SATISFAZÍVEL SE AO MENOS UMA DAS VARIÁVEIS EM CADA CLÁUSULA É VERDADEIRA

Um dos exemplos mais comuns de aplicações do 3-SAT é o de encontrar cliques em um dado grafo.

O grafo tendo n nodos, onde cada um é um dos literais da fórmula proposicional, podemos descobrir se existe um n-clique neste grafo apenas verificando a satisfazibilidade da fórmula.

Na imagem ao lado, o grafo demonstra a fórmula que é FNC com cada grupo vermelho como cláusula e os vértices verdes formam um clique (dado que x = falso e y = verdadeiro).



➔ 3-SAT:

- FORMA NORMAL CONJUNTIVA
- MÁXIMO DE TRÊS VARIÁVEIS EM CADA CLÁUSULA

EXATAMENTE- 1 3-SAT

Determina se existe satisfação em uma dada fórmula proposicional de maneira semelhante ao 3-SAT, da onde se origina, porém aqui uma das três variáveis em cada cláusula tem de ser verdadeira e as duas outras tem de ser obrigatoriamente falsas.

➔ 3-SAT:

- FORMA NORMAL CONJUNTIVA
- MÁXIMO DE TRÊS VARIÁVEIS EM CADA CLÁUSULA

EXATAMENTE- 1 3-SAT

Determina se existe satisfação em uma dada fórmula proposicional de maneira semelhante ao 3-SAT, da onde se origina, porém aqui uma das três variáveis em cada cláusula tem de ser verdadeira e as duas outras tem de ser obrigatoriamente falsas.

NÃO-IGUAIS 3-SAT

Outra variação do 3-SAT, aqui o problema a ser resolvido é a existência de valoração para as variáveis da forma de modo que dentro de nenhuma cláusula as variáveis tenham o mesmo valor verdadeiro.

➔ 3-SAT:

- FORMA NORMAL CONJUNTIVA
- MÁXIMO DE TRÊS VARIÁVEIS EM CADA CLÁUSULA

EXATAMENTE- 1 3-SAT

Determina se existe satisfação em uma dada fórmula proposicional de maneira semelhante ao 3-SAT, da onde se origina, porém aqui uma das três variáveis em cada cláusula tem de ser verdadeira e as duas outras tem de ser obrigatoriamente falsas.

NÃO-IGUAIS 3-SAT

Outra variação do 3-SAT, aqui o problema a ser resolvido é a existência de valoração para as variáveis da forma de modo que dentro de nenhuma cláusula as variáveis tenham o mesmo valor verdadeiro.

Estes casos são mais comumente usados para estudo e possuem menos aplicações interessantes de serem debatidas neste trabalho, sendo que são casos especiais do teorema de dicotomia de Schaefer, que dirá que qualquer problema que generalize o problema da satisfazibilidade booleana ou faz parte do grupo de complexidade de tempo polinomial, ou é NP-completo.