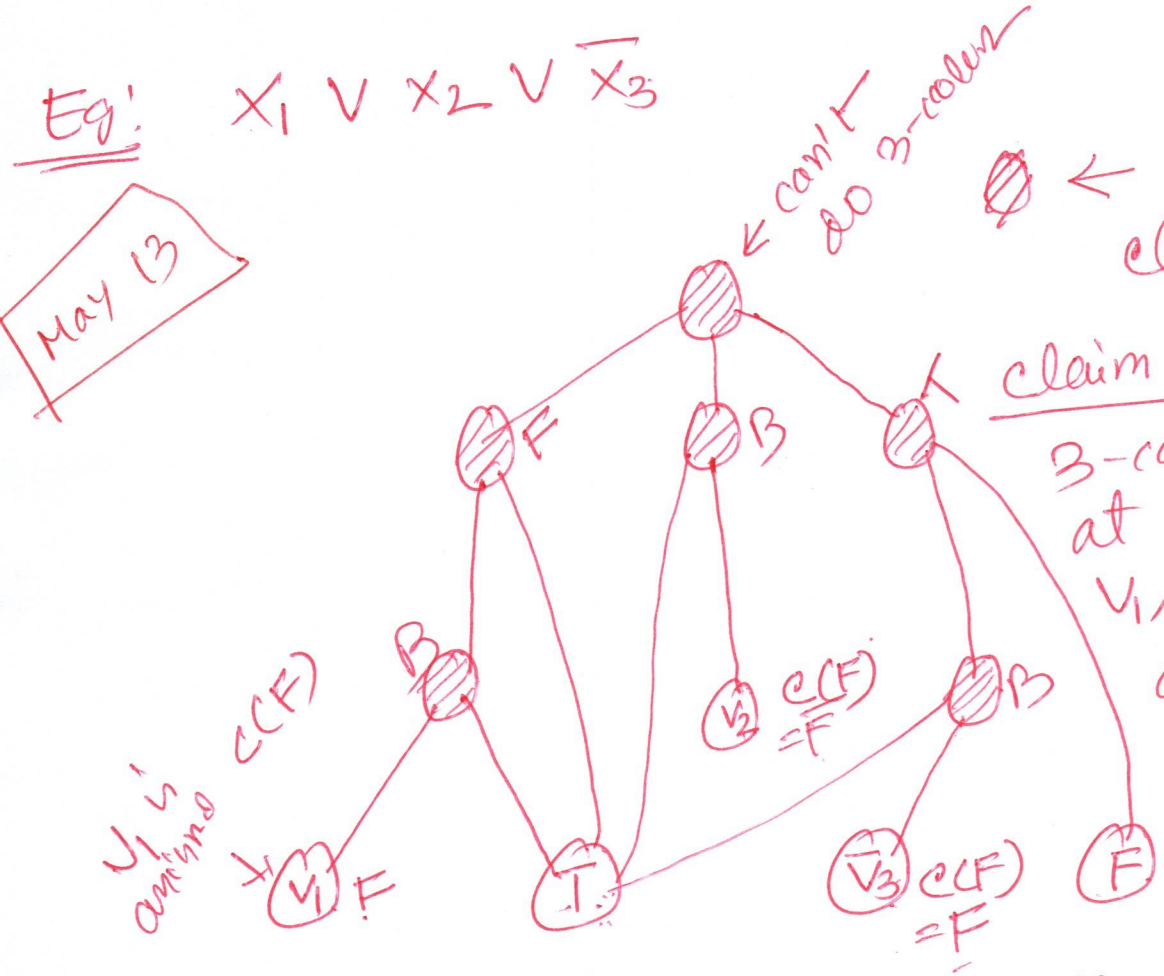
Step 2°: Encode each clause C_i with a gadget $Gadi$ that we'll "add" to G_0.

Eg: $x_1 \vee x_2 \vee \bar{x}_3$



claim°: In a valid 3-coloring of $Gadi$ at least one of v_1, v_2, and $\bar{v}_3$ is assigned C(T).

Pf: by contradict

$\Rightarrow$ if all v_1, v_2, $\bar{v}_3$ are colored C(F)
$\Rightarrow$ no valid 3-coloring of $Gadi$

$\exists$ a valid 3-coloring of $Gadi$ $\Leftrightarrow$ $\exists$ at least one literal in $Gadi$ is colored C(T)

Reduction: Given $\varphi = c_1, \ldots, c_m$ on $X = \{x_1, \ldots, x_n\}$

1. Compute G_0 on $\{v_1, \ldots, v_n, \bar{v}_1, \ldots, \bar{v}_n, T, F, B\}$
2. Add G_{c_i} to G_0 for each clause c_i
   ↳ let the resulting graph G_φ
3. Return Algo-3-color(G_φ)

THM: φ is satisfiable $\iff$ G_φ is 3-colorable.

NP-Hard "Harder" than NP-complete

Halting problem

i/P: a program/code P, an valid input I for P → (P, I)

O/P: 1 if P terminates/halts on I in finite time
0 o/w

Q: Does there ∃ an algo that solves the halting problem?

\# **THM**: NO!

Pf: By contradiction.

Assume $\exists$ an algo h that solves the halting problem.

$$\forall P, I, \quad h(P, I) = \begin{cases} 1, & \text{if } P(I) \text{ terminates} \\ 0, & \text{O/W} \end{cases}$$

def $C(x)$: ← x is a string

if $h(x, x) == 1$:

loop forever

else:

~~$h(x, x) = 0$~~

return

Call $C(C)$.

Case 1: if $h(C, C) = 1$ { $\Rightarrow C(C)$ terminates; $\Rightarrow C(C)$ loops forever.

Case 2: if $h(C, C) = 0$ { $\Rightarrow C(C)$ loops forever; $\Rightarrow C(C)$ terminates

↯ undecidability