

Outline

Exercise 3

Problem 1

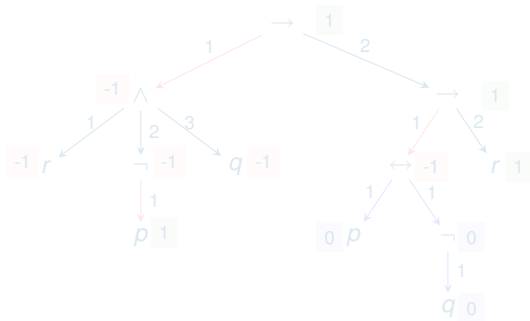
Problem 2

Problem 3

Problem 4

Exercise 3 (Problem 1)

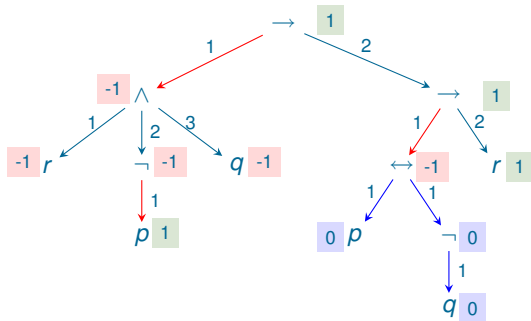
Draw the parse tree for the formula $r \wedge \neg p \wedge q \rightarrow ((p \leftrightarrow \neg q) \rightarrow r)$ and mark the nodes corresponding to the negative occurrences of subformulas (e.g., encircle them). Write down all negative subformulas of this formula.



$r \wedge \neg p \wedge q$
 r
 $\neg p$
 q
 $(p \leftrightarrow \neg q) \rightarrow r$

Exercise 3 (Problem 1)

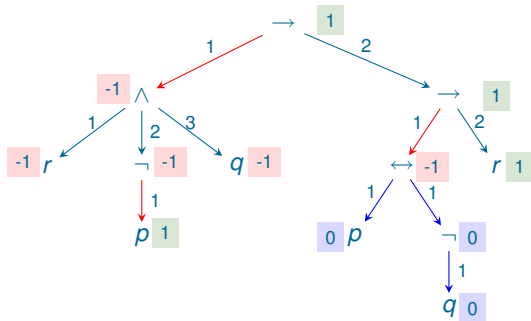
Draw the parse tree for the formula $r \wedge \neg p \wedge q \rightarrow ((p \leftrightarrow \neg q) \rightarrow r)$ and mark the nodes corresponding to the negative occurrences of subformulas (e.g., encircle them). Write down all negative subformulas of this formula.



$r \wedge \neg p \wedge q$
 r
 $\neg p$
 q
 $(p \leftrightarrow \neg q) \rightarrow r$

Exercise 3 (Problem 1)

Draw the parse tree for the formula $r \wedge \neg p \wedge q \rightarrow ((p \leftrightarrow \neg q) \rightarrow r)$ and mark the nodes corresponding to the negative occurrences of subformulas (e.g., encircle them). Write down all negative subformulas of this formula.



$r \wedge \neg p \wedge q$
 r
 $\neg p$
 q
 $(p \leftrightarrow \neg q) \rightarrow r$

Typical errors

- ▶ Nearly everybody parsed $r \wedge \neg p \wedge q$ as either $(r \wedge \neg p) \wedge q$ or $r \wedge (\neg p \wedge q)$.
- ▶ Some (very few) do not get the priorities right and ended up with a tree not rooted at \rightarrow .
- ▶ Some do not parse $\neg p$ further.
- ▶ In $p \leftrightarrow \neg q$, many decided that the occurrence of q has polarity -1.
- ▶ There are other mistakes with determining polarities, for example $r \wedge \neg p \wedge q$ occurs positively while its subformulas $r, \neg p, q$ negatively.

Exercise 3 (Problem 2)

Apply the definitional clausal form transformation algorithm (the non-optimized version) to the formula

$$\neg((p \rightarrow q) \leftrightarrow (\neg q \rightarrow \neg p))$$

	subformula	definition	clauses
			n_1
n_1	$\neg((p \rightarrow q) \leftrightarrow (\neg q \rightarrow \neg p))$	$n_1 \leftrightarrow \neg n_2$	$\neg n_1 \vee \neg n_2$ $n_1 \vee n_2$
n_2	$(p \rightarrow q) \leftrightarrow (\neg q \rightarrow \neg p)$	$n_2 \leftrightarrow (n_3 \leftrightarrow n_4)$	$\neg n_2 \vee \neg n_3 \vee n_4$ $\neg n_2 \vee \neg n_4 \vee n_3$ $n_3 \vee n_4 \vee n_2$ $\neg n_3 \vee \neg n_4 \vee n_2$
n_3	$p \rightarrow q$	$n_3 \leftrightarrow (p \rightarrow q)$	$\neg n_3 \vee \neg p \vee q$ $p \vee n_3$ $\neg q \vee n_3$
n_4	$\neg q \rightarrow \neg p$	$n_4 \leftrightarrow (\neg q \rightarrow \neg p)$	$\neg n_4 \vee q \vee \neg p$ $\neg q \vee n_4$ $p \vee n_4$

Exercise 3 (Problem 2)

Apply the definitional clausal form transformation algorithm (the non-optimized version) to the formula

$$\neg((p \rightarrow q) \leftrightarrow (\neg q \rightarrow \neg p))$$

	subformula	definition	clauses
			n_1
n_1	$\neg((p \rightarrow q) \leftrightarrow (\neg q \rightarrow \neg p))$	$n_1 \leftrightarrow \neg n_2$	$\neg n_1 \vee \neg n_2$ $n_1 \vee n_2$
n_2	$(p \rightarrow q) \leftrightarrow (\neg q \rightarrow \neg p)$	$n_2 \leftrightarrow (n_3 \leftrightarrow n_4)$	$\neg n_2 \vee \neg n_3 \vee n_4$ $\neg n_2 \vee \neg n_4 \vee n_3$ $n_3 \vee n_4 \vee n_2$ $\neg n_3 \vee \neg n_4 \vee n_2$
n_3	$p \rightarrow q$	$n_3 \leftrightarrow (p \rightarrow q)$	$\neg n_3 \vee \neg p \vee q$ $p \vee n_3$ $\neg q \vee n_3$
n_4	$\neg q \rightarrow \neg p$	$n_4 \leftrightarrow (\neg q \rightarrow \neg p)$	$\neg n_4 \vee q \vee \neg p$ $\neg q \vee n_4$ $p \vee n_4$

Typical errors

- ▶ .Many new variables for literals: they should only be introduced for formulas having at least one connective different from \neg .
- ▶ There were problems with transforming into CNF the formula $n_2 \leftrightarrow (n_3 \leftrightarrow n_4)$ (see the fourth row of the table).
- ▶ Many forgot to include the unit clause n_1 (second row of the table).
- ▶ Some applied standart CNF transformation instead of the definitional one.

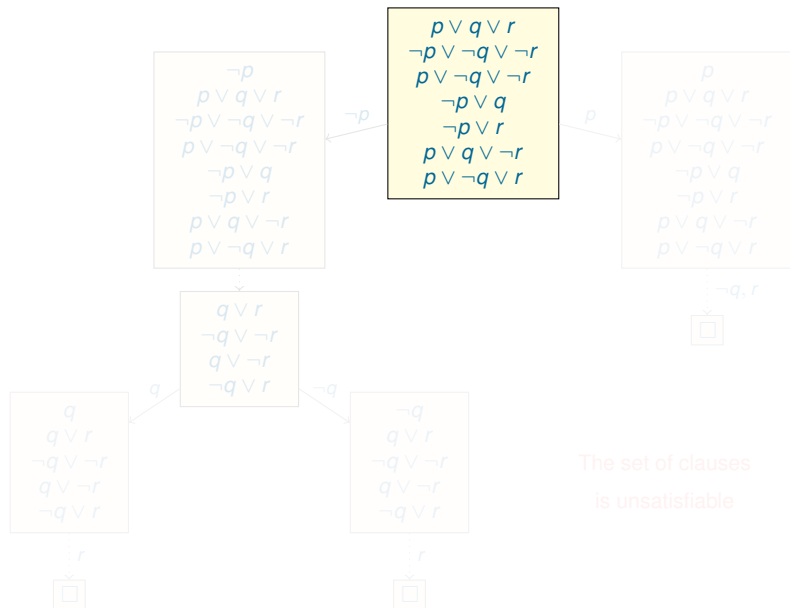
Exercise 3 (Problem 3)

Apply the DPLL algorithm to the following sets of clauses:

$$\begin{array}{l} p \vee q \vee r, \quad \neg p \vee \neg q \vee \neg r, \quad p \vee \neg q \vee \neg r, \quad \neg p \vee q, \\ \neg p \vee r, \quad p \vee q \vee \neg r, \quad p \vee \neg q \vee r. \end{array}$$

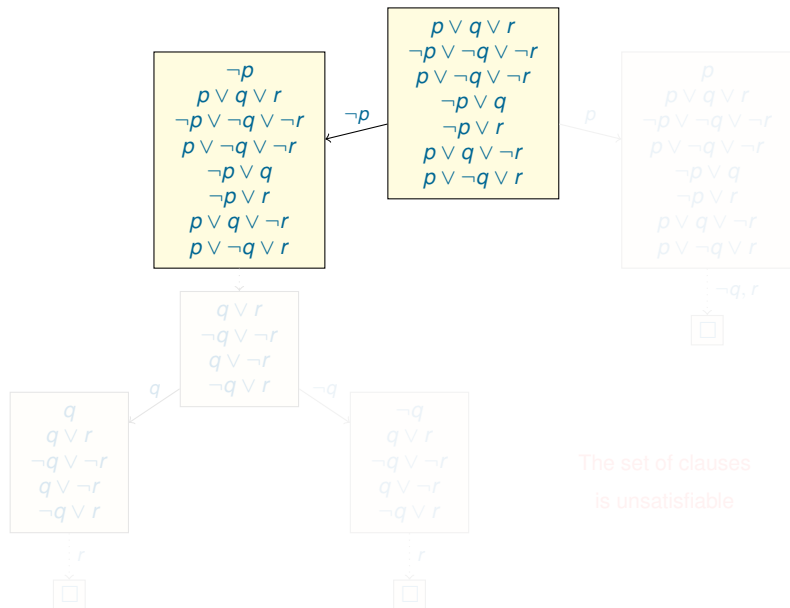
Is this set satisfiable? If yes, find a model of this set.

Solution

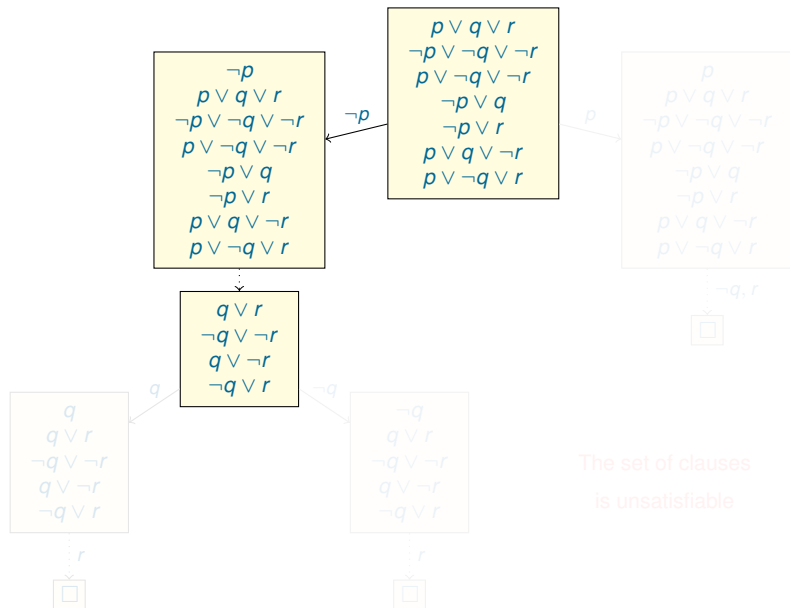


The set of clauses
is unsatisfiable

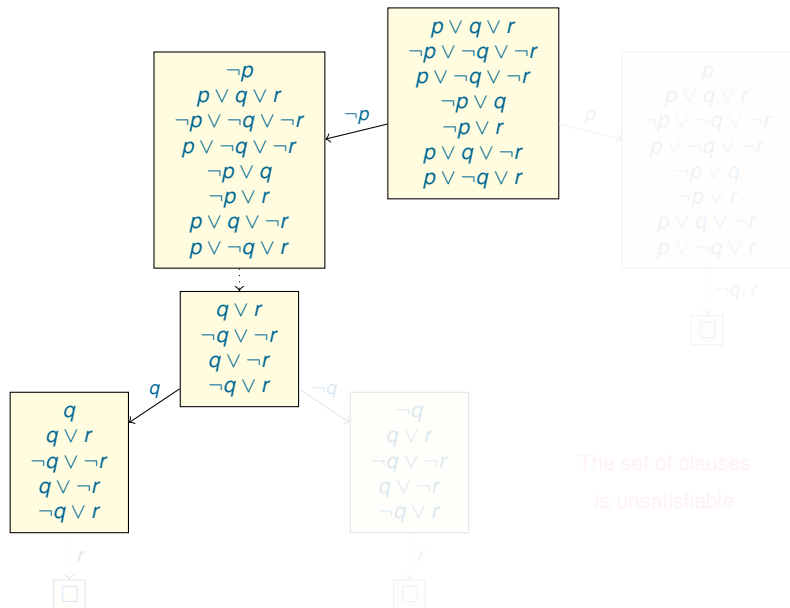
Solution



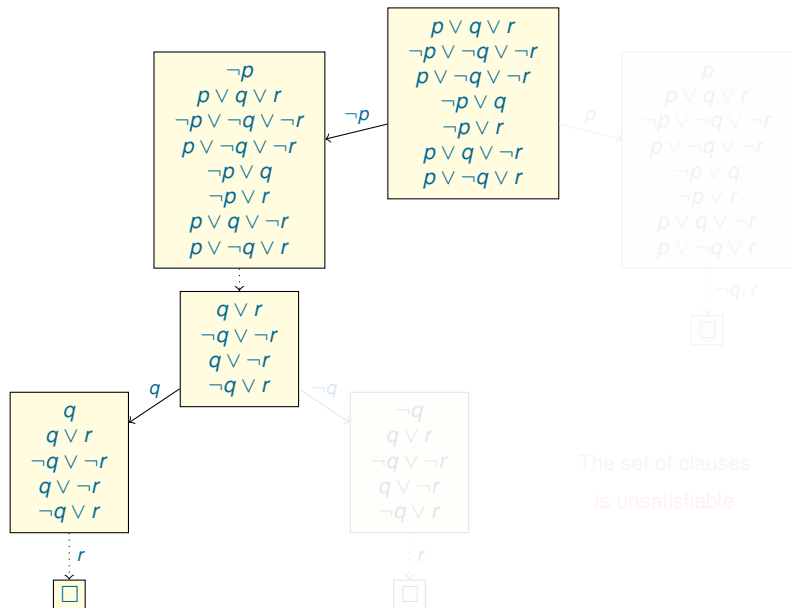
Solution



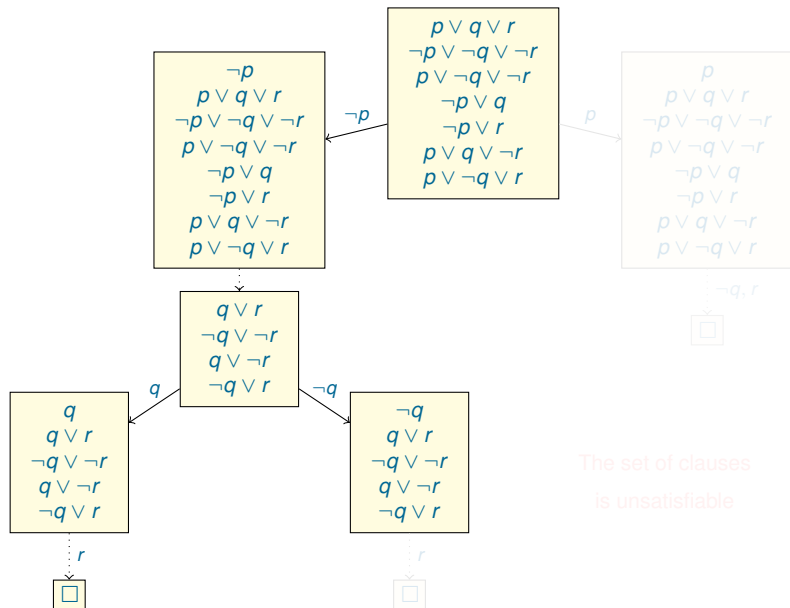
Solution



Solution

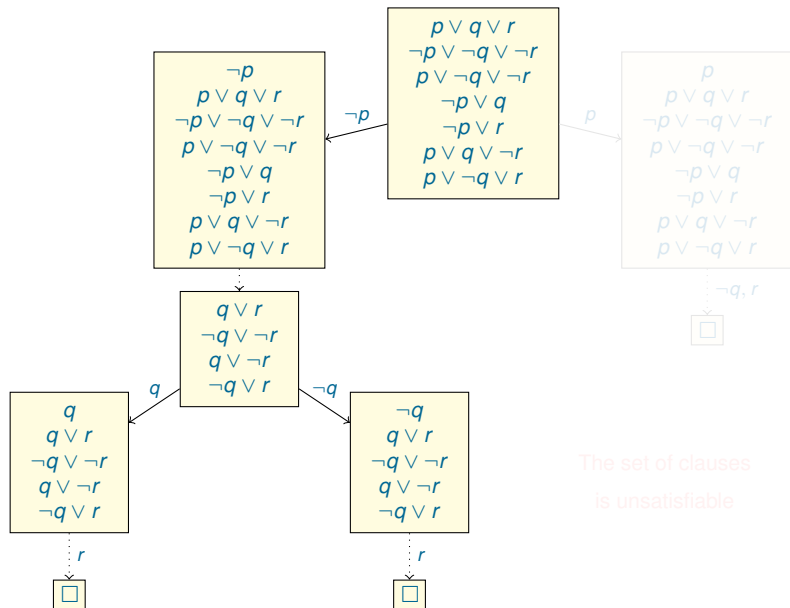


Solution

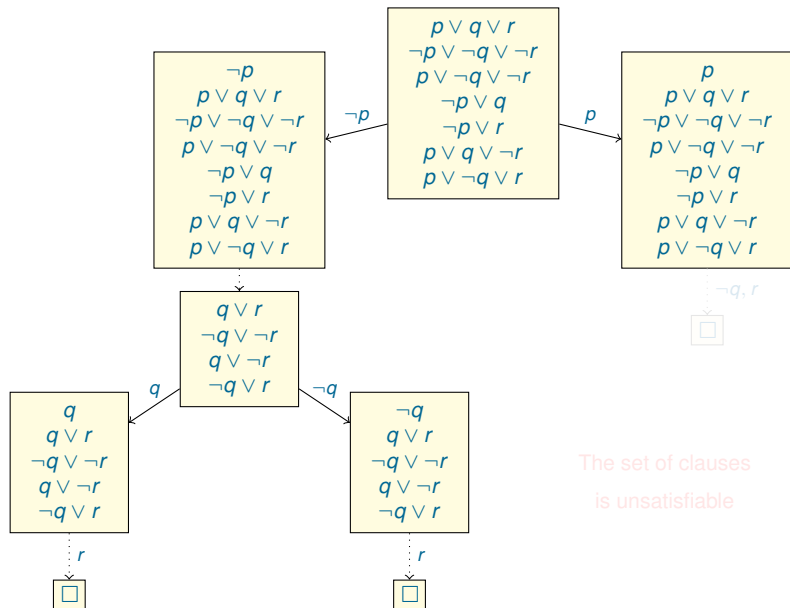


The set of clauses
is unsatisfiable

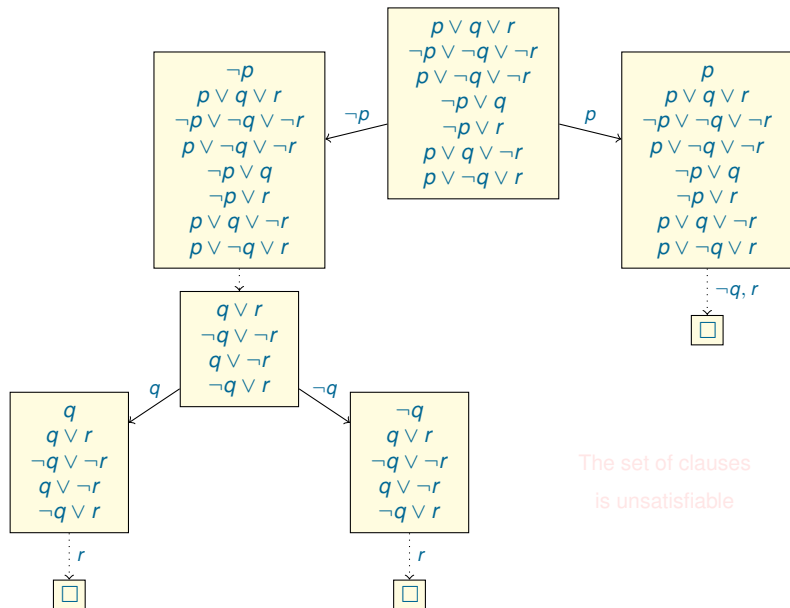
Solution



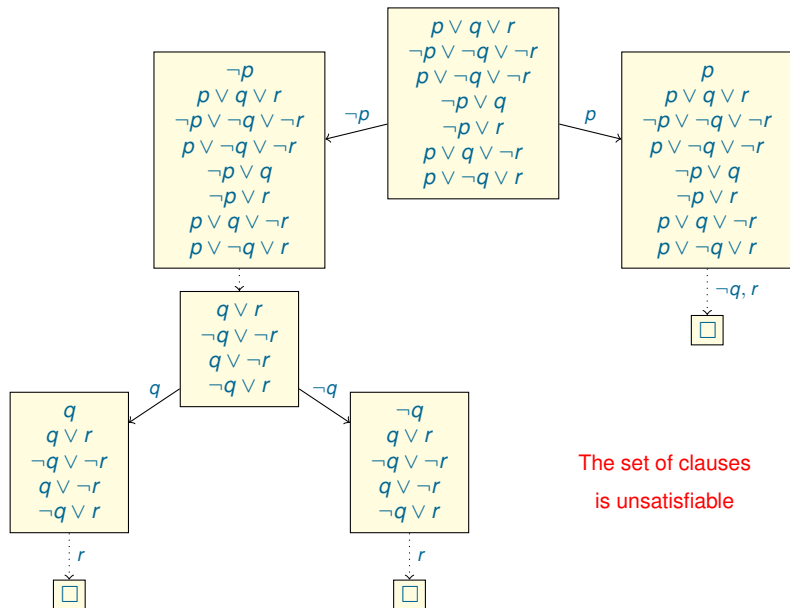
Solution



Solution



Solution



The set of clauses
is unsatisfiable

Typical errors

- ▶ Some built incomplete trees: when a model is not found, all branches should be exploited.
- ▶ Too many do not understand unit propagation, for example:
 - ▶ When a unit clause is derived branching is done instead of propagation.
 - ▶ Some write that a set is unsatisfiable but don't apply unit propagation.
 - ▶ Many do not delete clauses after propagating units;
 - ▶ Many do not remove the literal complementary to the one being propagated.
- ▶ Some don't use DPLL but simply say the set of clauses is satisfiable (or unsatisfiable).
- ▶ Some make very strange branching: one with q and the other with r .

Exercise 3 (Problem 4)

Apply the DPLL algorithm to the following sets of clauses:

$$\begin{array}{lll} p_1 \vee p_3, & \neg p_2 \vee \neg p_3, & p_1 \vee \neg p_3, \\ \neg p_1 \vee p_2, & p_1 \vee p_2 \vee \neg p_3, & p_1 \vee p_2 \vee p_3. \end{array}$$

Is this set satisfiable? If yes, find a model of this set.

Solution

$$\begin{array}{l} p_1 \vee p_3 \\ \neg p_2 \vee \neg p_3 \\ p_1 \vee \neg p_3 \\ \neg p_1 \vee p_2, \\ p_1 \vee p_2 \vee \neg p_3 \\ p_1 \vee p_2 \vee p_3 \end{array}$$

p_1

$$\begin{array}{l} p_1 \\ p_1 \vee p_3 \\ \neg p_2 \vee \neg p_3 \\ p_1 \vee \neg p_3 \\ \neg p_1 \vee p_2, \\ p_1 \vee p_2 \vee \neg p_3 \\ p_1 \vee p_2 \vee p_3 \end{array}$$

$p_2, \neg p_3$

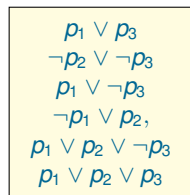
(empty set)

This set is **satisfiable**

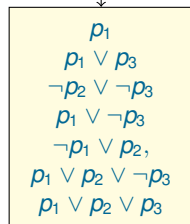
Its **model** can be obtained by collecting decision literals and propagated literals:

$$\{p_1 \mapsto 1, p_2 \mapsto 1, p_3 \mapsto 0\}$$

Solution



p_1



$p_2, \neg p_3$

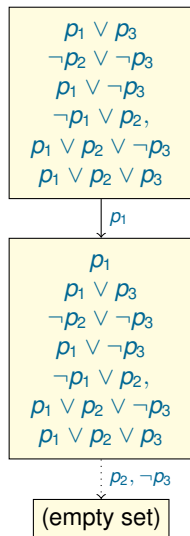
(empty set)

This set is **satisfiable**

Its **model** can be obtained by collecting decision literals and propagated literals:

$\{p_1 \mapsto 1, p_2 \mapsto 1, p_3 \mapsto 0\}$

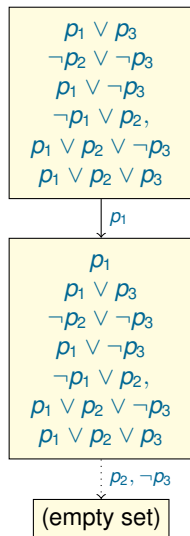
Solution



This set is **satisfiable**

Its **model** can be obtained by collecting decision literals and propagated literals:
 $\{p_1 \mapsto 1, p_2 \mapsto 1, p_3 \mapsto 0\}$

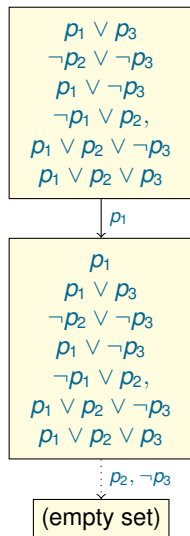
Solution



This set is **satisfiable**

Its **model** can be obtained by collecting decision literals and propagated literals:
 $\{p_1 \mapsto 1, p_2 \mapsto 1, p_3 \mapsto 0\}$

Solution



This set is **satisfiable**

Its **model** can be obtained by collecting decision literals and propagated literals:

$$\{p_1 \mapsto 1, p_2 \mapsto 1, p_3 \mapsto 0\}$$

Typical errors

- ▶ Some create three branches coming out: with p_1 , p_2 and p_3 as units.
- ▶ Some have difficulty to illustrate the process of running DPLL just crossing deleted clauses and variables, and lose a branch as a result.
- ▶ Some give a model in which a variable is undefined (all variables should be defined in a model).
- ▶ Sometimes a model is given as a set of literals $\{p_1, p_2, \neg p_3\}$ (a model should map variables to values).
- ▶ Some take a strange set of clauses as initial and run DPLL on it instead of the set of clauses from the question.
- ▶ Some do not understand what an empty clause is and do not understand the difference between the empty clause and the empty set of clauses.