Propositional Logic IV

Zhaoguo Wang

Adapted From:

NYU G22.2390-001, Propositional Logic
教材《数理逻辑与集合论》1.4, 2.6
<<Solving SAT and SAT Modulo Theories: From an Abstract Davis–Putnam–Logemann–Loveland Procedure to DPLL(T)>>

Propositional Logic（命题逻辑）

1.1 Propositional Logic

Step 0. Proof.

Step 1. Convert program into mathematical formula.

Step 2. Ask the computer to solve the formula.

1.2 First Order Logic

Step 1. Convert it into first logic formula.

Step 2. Ask the computer to solve the formula.

1.3 Interactive Proof

Step 1. Axiom system

Step 2. Ask the computer to check the invariants

A Quick Recap – Last Lecture

- Tautology
- SAT Problem

Outline – This Lecture

- Normal Form
- DPLL

SAT（可满足性问题）

DEFINITION

In computer science, the satisfiability problem (abbreviated SATISFIABILITY or SAT) is the problem of determining if there exists an interpretation that satisfies a given formula.

The first problem that was proven to be NP-complete (Cook-Levin theorem)

SAT（可满足性问题）

DEFINITION

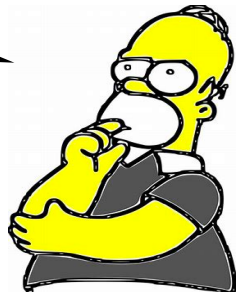
In computer science, the satisfiability problem (abbreviated SATISFIABILITY or SAT) is the problem of determining if there exists an interpretation that satisfies a given formula.

S. A. Cook.
The Complexity of Theorem Proving Procedures.
Proceedings of the Third Annual ACM Symposium on the Theory of Computing, 151-158, 1971.

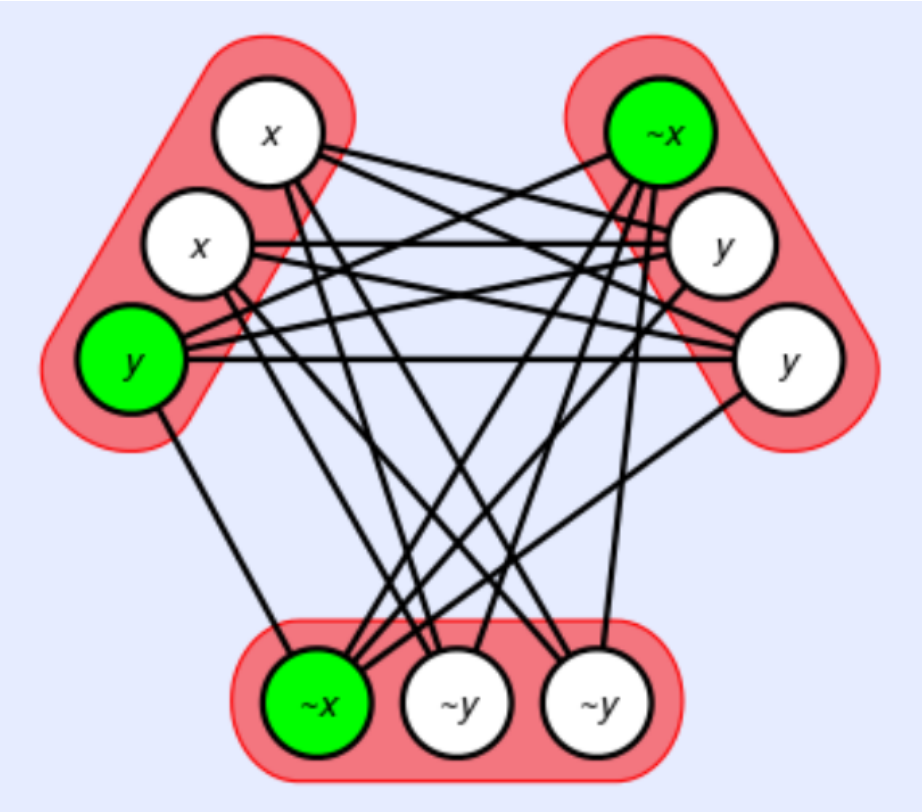
Stephen Cook

Leonid Levin

Is there a way to speed up the algorithm?



SAT solver can do it!



http://www.satcompetition.org/

Normal Form（范式）

OBSERVATION

Given a wff, you can build infinite formulas that are equivalent to it.

$$A \to B, \neg A \vee B, \neg(A \wedge \neg B)$$

They are in different form
but they are equivalent.

Normal Form（范式）

OBSERVATION

Given a wff, you can build infinite formulas that are equivalent to it.

$$A \to B, \neg A \vee B, \neg(A \wedge \neg B)$$

If we can require a normal form, it is good for designing algorithm for SAT problem.

Normal Form（范式）

DEFINITION

A literal（文字）is a propositional variable or its negation.

A clause（析取式/子句）is a disjunction of one or more literals.

A conjunctive clause（合取式） is a conjunction of one or more literals.

A	literal
$\neg B$	literal
$\neg A \vee B$	disjunctive clause
$A \wedge \neg B$	conjunctive clause

Conjunction Normal Form（合取范式）

DEFINITION

A formula is in conjunctive normal form (CNF) or clausal normal form if it is a conjunction of one or more clauses. Otherwise put, it is an AND of ORs.

Modern SAT solvers use CNF.

$$(A \vee \neg B) \wedge (B \vee \neg A) \wedge A$$

$$A \wedge B$$

Converting to CNF

However, many problems are not in CNF. How to convert them to CNF?

Converting to CNF

P	Q	$g_0(P, Q)$	$g_1(P, Q)$
F	F	T	T
F	T	T	T
T	F	F	F
T	T	T	F

Step1: find all false rows

Step2: generate a formula for every row

Step3: use "and" to connect these formulas

$$g_1(P, Q) = ((\neg P) \vee Q) \wedge ((\neg P) \vee (\neg Q))$$

It is in CNF! Truth table can help convert formulas to CNF.

Converting to CNF

P	Q	$g_0(P, Q)$	$g_1(P, Q)$
F	F	T	T
F	T	T	T
T	F	F	F
T	T	T	F

Step1: find all false rows

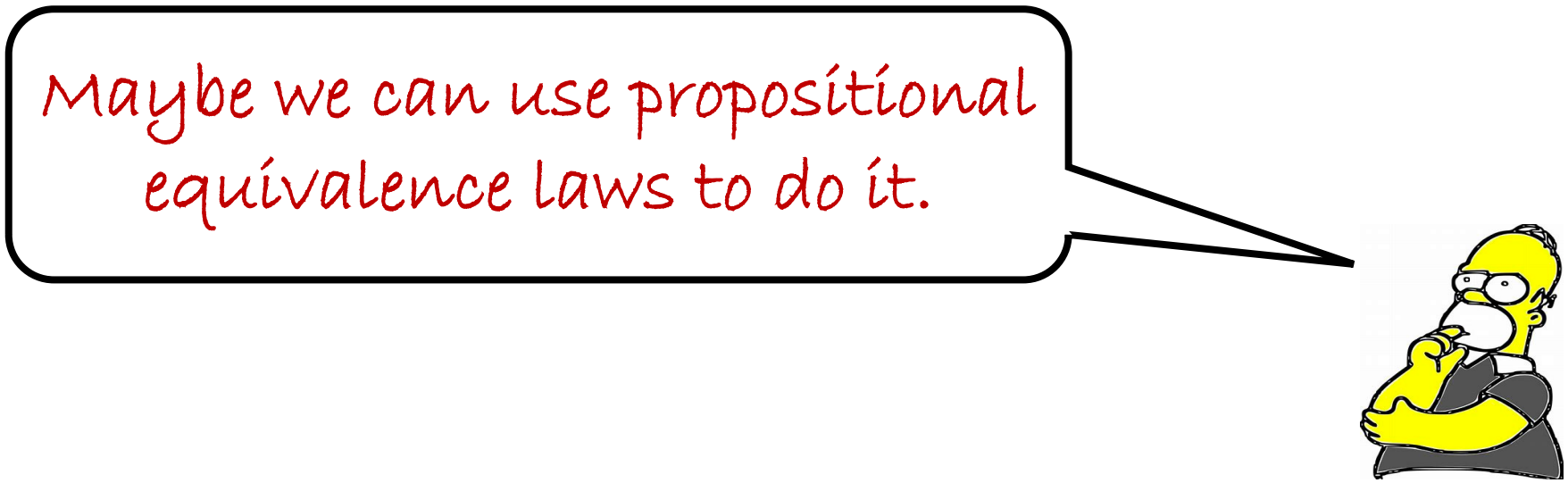
Step2: generate a formula for every row

Step3: use "and" to connect these formulas

$$g_1(P, Q) = ((\neg P) \vee Q) \wedge ((\neg P) \vee (\neg Q))$$

But if we can build the truth table, we don't need SAT solvers.

Converting to CNF



Converting to CNF

$$\neg(P \vee Q) \leftrightarrow (P \wedge Q)$$

Example

Converting to CNF

$$\neg(P \vee Q) \leftrightarrow (P \wedge Q)$$

$$=(\neg\neg(P \vee Q) \vee (P \wedge Q)) \wedge (\neg(P \vee Q) \vee \neg(P \wedge Q))$$

Step1: eliminate implication and biconditional connectives.

Converting to CNF

$$\neg(P \vee Q) \leftrightarrow (P \wedge Q)$$
$$=(\neg\neg(P \vee Q) \vee (P \wedge Q)) \wedge (\neg(P \vee Q) \vee \neg(P \wedge Q))$$
$$=((P \vee Q) \vee (P \wedge Q)) \wedge ((\neg P \wedge \neg Q) \vee (\neg P \vee \neg Q))$$

Step1: eliminate implication and biconditional connectives.

Step2: repeatedly use De Morgan's laws and double negation law to move "not" inside

Converting to CNF

$$\neg(P \vee Q) \leftrightarrow (P \wedge Q)$$
$$=(\neg\neg(P \vee Q) \vee (P \wedge Q)) \wedge (\neg(P \vee Q) \vee \neg(P \wedge Q))$$
$$=((P \vee Q) \vee (P \wedge Q)) \wedge ((\neg P \wedge \neg Q) \vee (\neg P \vee \neg Q))$$
$$=(P \vee Q) \wedge (\neg P \vee \neg Q)$$

Step1: eliminate implication and biconditional connectives.

Step2: repeatedly use De Morgan's laws and double negation law to move "not" inside

Step3: repeatedly use distributive law to get CNF

Exercise

$$(A \wedge B) \leftrightarrow E$$

$$(A \wedge B) \leftrightarrow E$$
$$((A \wedge B) \rightarrow E) \wedge (E \rightarrow (A \wedge B))$$
$$(\neg(A \wedge B) \vee E) \wedge (\neg E \vee (A \wedge B))$$
$$(\neg A \vee \neg B \vee E) \wedge (\neg E \vee A) \wedge (\neg E \vee B)$$

Are there any other normal forms?

Disjunction Normal Form（析取范式）

DEFINITION

A formula is in disjunctive normal form (DNF) if it is a disjunction of one or more conjunction clauses. Otherwise put, it is an OR of ANDs.

$$A \vee B$$

$$(A \wedge B) \vee (\neg A \wedge \neg B)$$

Converting to DNF

$$\neg(P \vee Q) \leftrightarrow (P \wedge Q)$$

Same example in CNF

Converting to DNF

$$\neg(P \vee Q) \leftrightarrow (P \wedge Q)$$

$$=(\neg\neg(P \vee Q) \wedge \neg(P \wedge Q)) \vee (\neg(P \vee Q) \wedge (P \wedge Q))$$

Step1: eliminate implication and biconditional connectives.

Converting to DNF

$$\neg(P \vee Q) \leftrightarrow (P \wedge Q)$$
$$=(\neg\neg(P \vee Q) \wedge \neg(P \wedge Q)) \vee (\neg(P \vee Q) \wedge (P \wedge Q))$$
$$=((P \vee Q) \wedge (\neg P \vee \neg Q)) \vee (\neg P \wedge \neg Q \wedge P \wedge Q)$$

Step1: eliminate implication and biconditional connectives.

Step2: repeatedly use De Morgan's laws and double negation law to move "not" inside

Converting to DNF

Step1: eliminate implication and biconditional connectives.

$$\neg(P \vee Q) \leftrightarrow (P \wedge Q)$$
$$=(\neg\neg(P \vee Q) \wedge \neg(P \wedge Q)) \vee (\neg(P \vee Q) \wedge (P \wedge Q))$$
$$=((P \vee Q) \wedge (\neg P \vee \neg Q)) \vee (\neg P \wedge \neg Q \wedge P \wedge Q)$$
$$=(P \wedge \neg P) \vee (Q \wedge \neg P) \vee (P \wedge \neg Q) \vee (Q \wedge \neg Q) \vee (\neg P \wedge \neg Q \wedge P \wedge Q)$$

Step2: repeatedly use De Morgan's laws and double negation law to move "not" inside

Step3: repeatedly use distributive law to get DNF

Exercise

$$(A \wedge B) \leftrightarrow E$$

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

Why does SAT Solvers use CNF instead of DNF?
If SAT solvers use DNF, what will happen?

CNF v.s. DNF

$$A_1 \wedge A_2 \wedge A_3 \wedge \ldots \wedge A_n$$

$$A_1 \vee A_2 \vee A_3 \vee \ldots \vee A_n$$

A formula in DNF is true when one of these conjunction clauses is true. One of them is true when P and $\neg P$ don't exist at the same time.

CNF v.s. DNF

$$A_1 \wedge A_2 \wedge A_3 \wedge \ldots \wedge A_n$$

$$A_1 \vee A_2 \vee A_3 \vee \ldots \vee A_n$$

You just have to scan through the conjunction clauses and check whether one of them contains not both a variable and its negation.

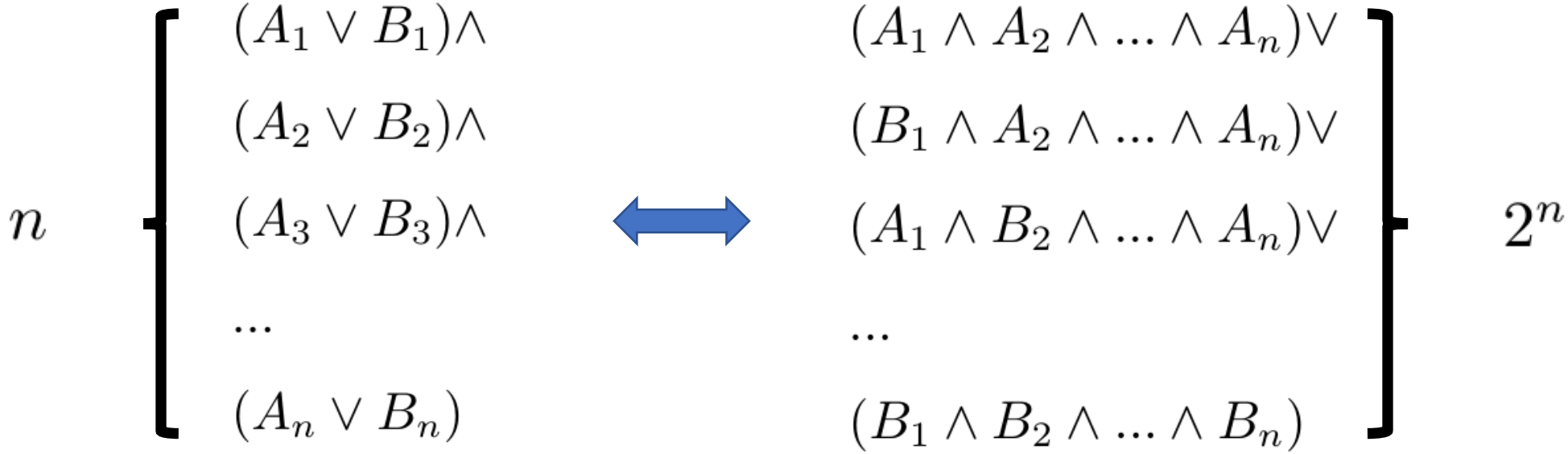
CNF v.s. DNF

$$A_1 \wedge A_2 \wedge A_3 \wedge \ldots \wedge A_n$$

$$A_1 \vee A_2 \vee A_3 \vee \ldots \vee A_n$$

Things become much easier when we address DNF instead of CNF. Why SAT solvers use CNF?

Disjunction Normal Form



You can convert a formula into DNF, but the resulting formula might be very much larger than the original formula—in fact, exponentially so.

Normal Form Theorem

THEOREM

Every formula can be converted to CNF and DNF.

We can use truth table to convert a formula to CNF and DNF.

$$\neg(P \vee Q) \leftrightarrow (P \wedge Q)$$

$$(P \vee Q) \wedge (\neg P \vee \neg Q)$$

$$(P \vee Q) \wedge (\neg P \vee \neg Q) \wedge (P \vee \neg P)$$

Although we can convert formulas to CNF, we can still get many different formulas. Can we define a more precise form?

Principal Normal Form（主范式）

DEFINITION

Given n propositional variables, if every variable exists once in a conjunction clause, the clause is a minterm（极小项）.

$P1 \;\; P2$ ➡ $P1 \wedge P2, \neg P1 \wedge P2, P1 \wedge \neg P2, \neg P1 \wedge \neg P2$ $\quad 2^n$

Every minterm is true under only one interpretation. Any two minterms are not equivalent and the conjunction of them is F.

Principal Disjunction Normal Form（主析取范式）

DEFINITION

Principal Disjunction Normal Form(PDNF) is the disjunction of minterms.

P	Q	$g_0(P,Q)$	$g_1(P,Q)$
F	F	T	T
F	T	T	T
T	F	F	F
T	T	T	F

Step1: find all true rows

Step2: generate a formula for every row

Step3: use "or" to connect these formulas

$$g_0(P,Q) = (((\neg P) \wedge (\neg Q)) \vee ((\neg P) \wedge Q)) \vee (P \wedge Q)$$

Truth table is all-purpose. Every formula con be converted to only one PDNF.

Principal Disjunction Normal Form（主析取范式）

DEFINITION

Principal Disjunction Normal Form(PDNF) is the disjunction of minterms.

P	Q	$g_0(P,Q)$	$g_1(P,Q)$
F	F	T	T
F	T	T	T
T	F	F	F
T	T	T	F

Step1: find all true rows

Step2: generate a formula for every row

Step3: use "or" to connect these formulas

$$g_0(P,Q) = (((\neg P) \wedge (\neg Q)) \vee ((\neg P) \wedge Q)) \vee (P \wedge Q)$$

If a PDNF includes all minterms, it must be true.

Converting to PDNF

$$
\begin{aligned}
& P \to Q \\
=& \neg P \vee Q \\
=& (\neg P \wedge (Q \vee \neg Q)) \vee Q \\
=& (\neg P \wedge Q) \vee (\neg P \wedge \neg Q) \vee Q \\
=& (\neg P \wedge Q) \vee (\neg P \wedge \neg Q) \vee (Q \wedge (P \vee \neg P)) \\
=& (\neg P \wedge Q) \vee (\neg P \wedge \neg Q) \vee (P \wedge Q) \vee (\neg P \wedge Q) \\
=& (\neg P \wedge \neg Q) \vee (\neg P \wedge Q) \vee (P \wedge Q) \\
=& m_0 \vee m_1 \vee m_3 = \vee_{0;1;3}
\end{aligned}
$$

Exercise

$$(A \wedge B) \leftrightarrow E$$

$$
\begin{aligned}
&(A \wedge B) \leftrightarrow E \\
=&(A \wedge B \wedge E) \vee (\neg(A \wedge B) \wedge \neg E) \\
=&(A \wedge B \wedge E) \vee ((\neg A \vee \neg B) \wedge \neg E) \\
=&(A \wedge B \wedge E) \vee (\neg A \wedge \neg E) \vee (\neg B \wedge \neg E) \\
=&(A \wedge B \wedge E) \vee (\neg A \wedge \neg E \wedge (B \vee \neg B)) \vee (\neg B \wedge \neg E \wedge (A \vee \neg A)) \\
=&(A \wedge B \wedge E) \vee (\neg A \wedge B \wedge \neg E) \vee (\neg A \wedge \neg B \wedge \neg E) \vee (A \wedge \neg B \wedge \neg E) \vee (\neg A \wedge \neg B \wedge \neg E) \\
=&(\neg A \wedge \neg B \wedge \neg E) \vee (\neg A \wedge B \wedge \neg E) \vee (A \wedge \neg B \wedge \neg E) \vee (A \wedge B \wedge E) \\
=&m_0 \vee m_2 \vee m_4 \vee m_7 = \vee_{0;2;4;7}
\end{aligned}
$$

Principal Normal Form（主范式）

DEFINITION

Given n propositional variables, if every variable exists once in a disjunction clause, the clause is a maxterm（极大项）.

$P1 \;\; P2 \Rightarrow P1 \vee P2, \neg P1 \vee P2, P1 \vee \neg P2, \neg P1 \vee \neg P2$ $\qquad 2^n$

Every maxterm is false under only one interpretation.
Every two maxterms are not equivalent and the disjunction of them is T.

Principal Conjunction Normal Form（主合取范式）

DEFINITION

Principal Conjunction Normal Form(PCNF) is the conjunction of maxterms.

P	Q	$g_0(P,Q)$	$g_1(P,Q)$
F	F	T	T
F	T	T	T
T	F	F	F
T	T	T	F

Step1: find all false rows

Step2: generate a formula for every row

Step3: use "and" to connect these formulas

$$g_1(P,Q) = ((\neg P) \vee Q) \wedge ((\neg P) \vee (\neg Q))$$

Truth table is all-purpose. Every formula con be converted to only one PCNF.

Principal Conjunction Normal Form（主合取范式）

DEFINITION

Principal Conjunction Normal Form(PCNF) is the conjunction of maxterms.

P	Q	$g_0(P,Q)$	$g_1(P,Q)$
F	F	T	T
F	T	T	T
T	F	F	F
T	T	T	F

Step1: find all false rows

Step2: generate a formula for every row

Step3: use "and" to connect these formulas

$$g_1(P,Q) = ((\neg P) \vee Q) \wedge ((\neg P) \vee (\neg Q))$$

If a PCNF includes all maxterms, it must be false.

Converting to PCNF

$$
\begin{aligned}
& P \wedge Q \\
=& (P \vee (Q \wedge \neg Q)) \wedge (Q \vee (P \wedge \neg P)) \\
=& (P \vee Q) \wedge (P \vee \neg Q) \wedge (Q \vee P) \wedge (Q \vee \neg P) \\
=& (\neg P \vee Q) \wedge (P \vee \neg Q) \wedge (P \vee Q) \\
=& M_1 \wedge M_2 \wedge M_3 = \wedge_{1;2;3}
\end{aligned}
$$

Exercise

$$(A \wedge B) \leftrightarrow E$$

$$
\begin{aligned}
&(A \wedge B) \leftrightarrow E \\
=&(A \wedge B \rightarrow E) \wedge (E \rightarrow A \wedge B) \\
=&(\neg(A \wedge B) \vee E) \wedge (\neg E \vee (A \wedge B)) \\
=&(\neg A \vee \neg B \vee E) \wedge (\neg E \vee A) \wedge (\neg E \vee B) \\
=&(\neg A \vee \neg B \vee E) \wedge (\neg E \vee A \vee (B \wedge \neg B)) \wedge (\neg E \vee B \vee (A \wedge \neg A)) \\
=&(\neg A \vee \neg B \vee E) \wedge (A \vee B \vee \neg E) \wedge (A \vee \neg B \vee \neg E) \wedge (A \vee B \vee \neg E) \wedge (\neg A \vee B \vee \neg E) \\
=&(\neg A \vee \neg B \vee E) \wedge (\neg A \vee B \vee \neg E) \wedge (A \vee \neg B \vee \neg E) \wedge (A \vee B \vee \neg E) \\
=&M_1 \wedge M_2 \wedge M_4 \wedge M_6 = \wedge_{1;2;4;6}
\end{aligned}
$$

PCNF vs. PDNF

P	Q	$g_0(P,Q)$	$g_1(P,Q)$
F	F	T	T
F	T	T	T
T	F	F	F
T	T	T	F

P	Q	$g_0(P,Q)$	$g_1(P,Q)$
F	F	T	T
F	T	T	T
T	F	F	F
T	T	T	F

$$g_0(P,Q) = (((\neg P) \wedge (\neg Q)) \vee ((\neg P) \wedge Q)) \vee (P \wedge Q)$$

$$g_1(P,Q) = ((\neg P) \wedge (\neg Q)) \vee ((\neg P) \wedge Q)$$

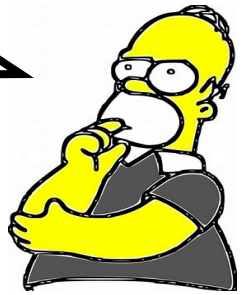
$$g_0(P,Q) = ((\neg P) \vee Q)$$

$$g_1(P,Q) = ((\neg P) \vee Q) \wedge ((\neg P) \vee (\neg Q))$$

Can you find the relation between PCNF and PDNF according to truth table?

SAT Solver

In practice, CNF is enough and more convenient for SAT Solver.



Strawman

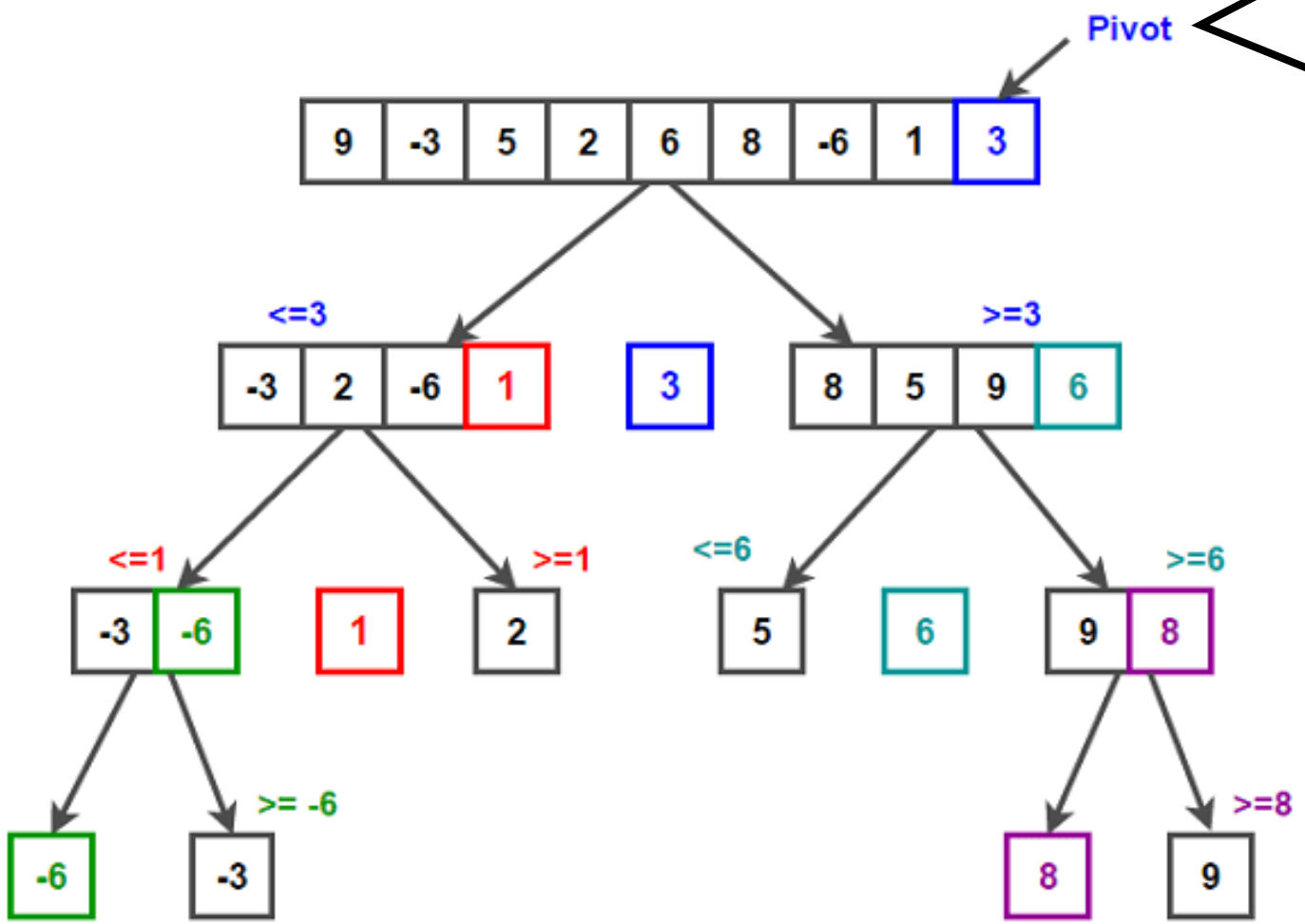
RROBLEM

Given a CNF formula, check its satisfiability efficiently

BASIC IDEA

Randomly assign the truth value to ***maximize the number of true clauses.***

Algorithms Using Randomness



Select the pivot randomly in quicksort usually bring better performance and avoid worst case.

Strawman

RROBLEM

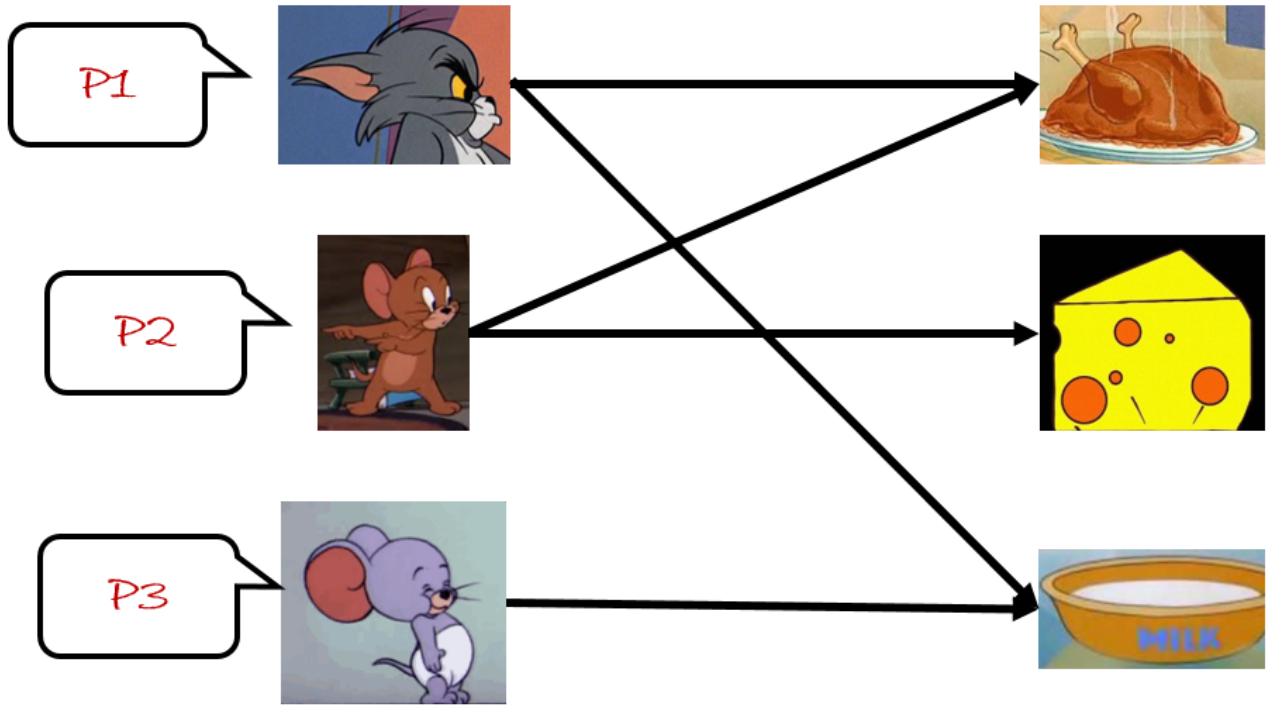
Given a CNF formula, check its satisfiability efficiently

GSAT(random + greedy)

Loop n times

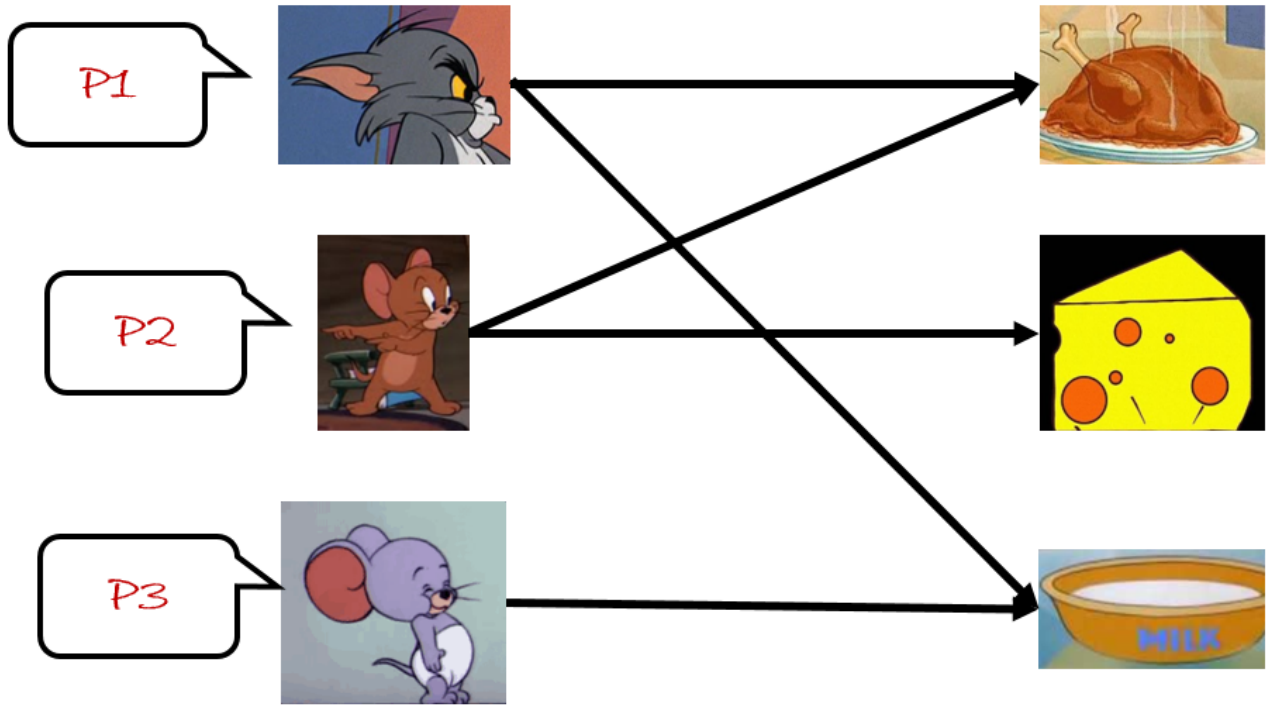
- Randomly select start assignment of variables
- Loop m times
  - Flip a variable that results in maximum profit
  - Exit if profit = number of clauses

Example



$$3,\quad \overline{1} \vee \overline{2},\quad 1 \vee \overline{3}$$

Example



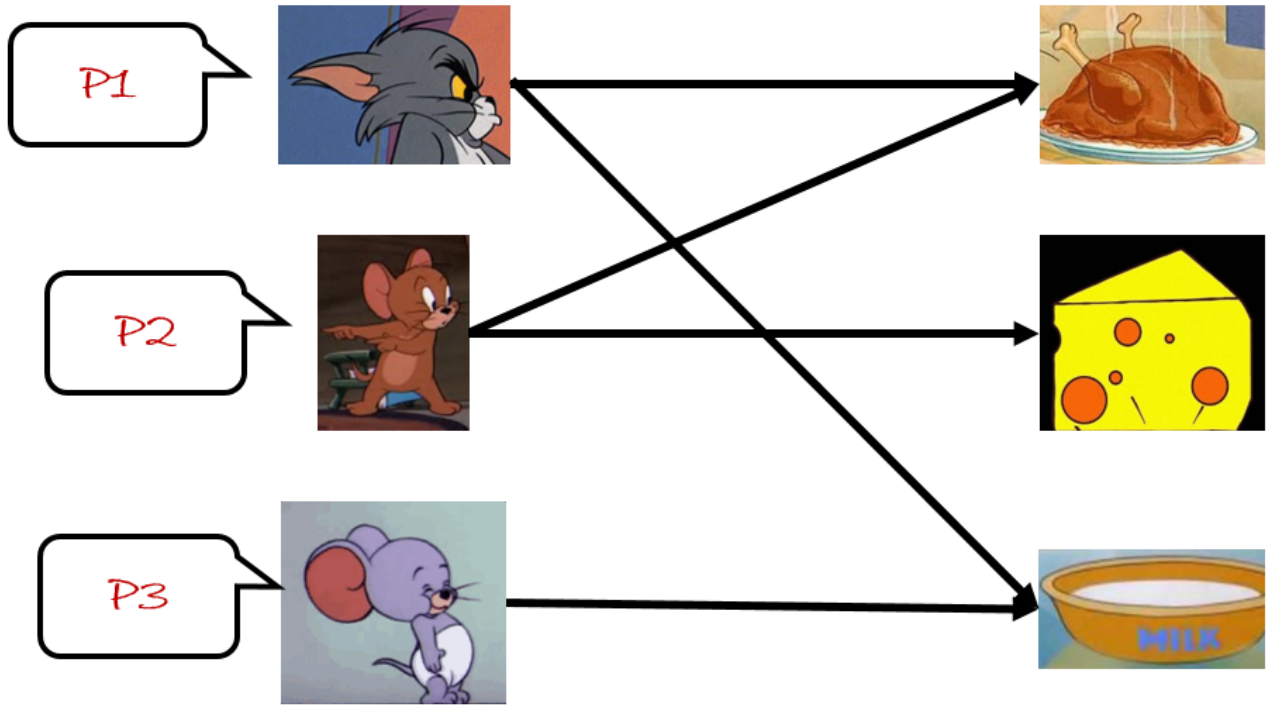
$$3,\ \overline{1} \vee \overline{2},\ 1 \vee \overline{3}$$

starting point

1	2	3	profit
T	T	F	1
T	F	F	2
T	F	T	3

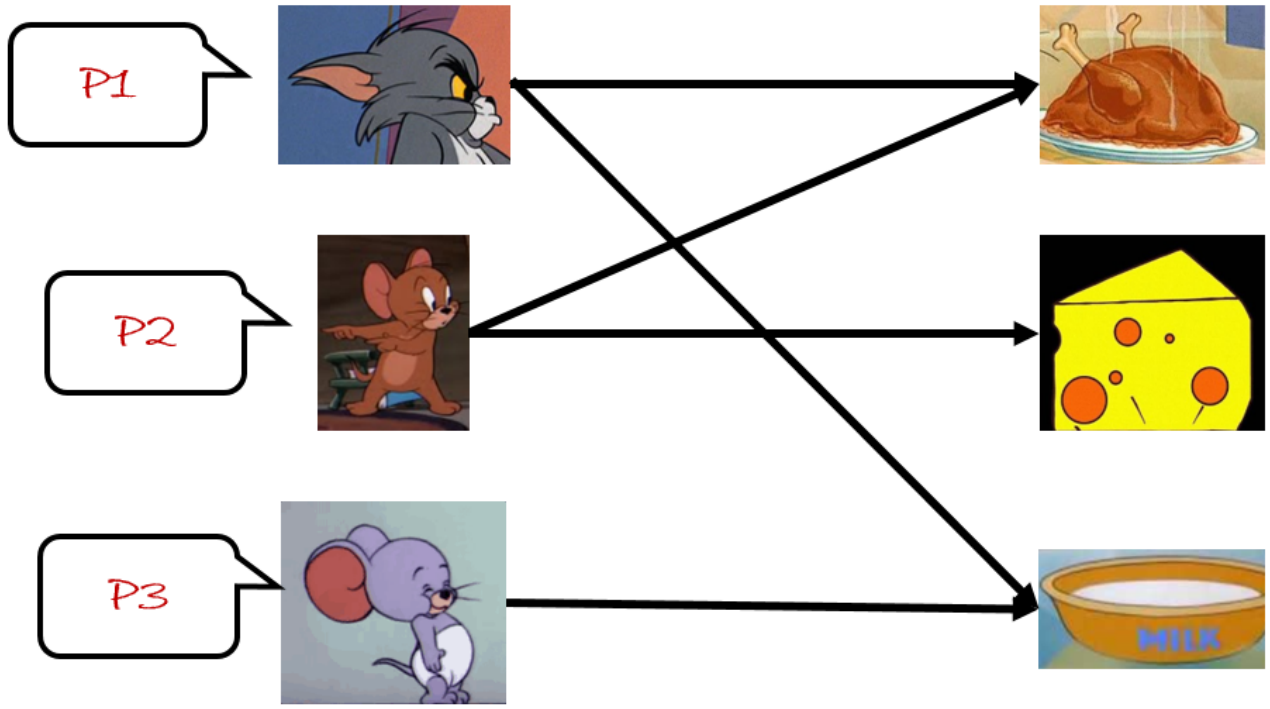
success

Issue



$$3,\quad \overline{1} \vee \overline{2},\quad 1 \vee \overline{3}$$

Issue



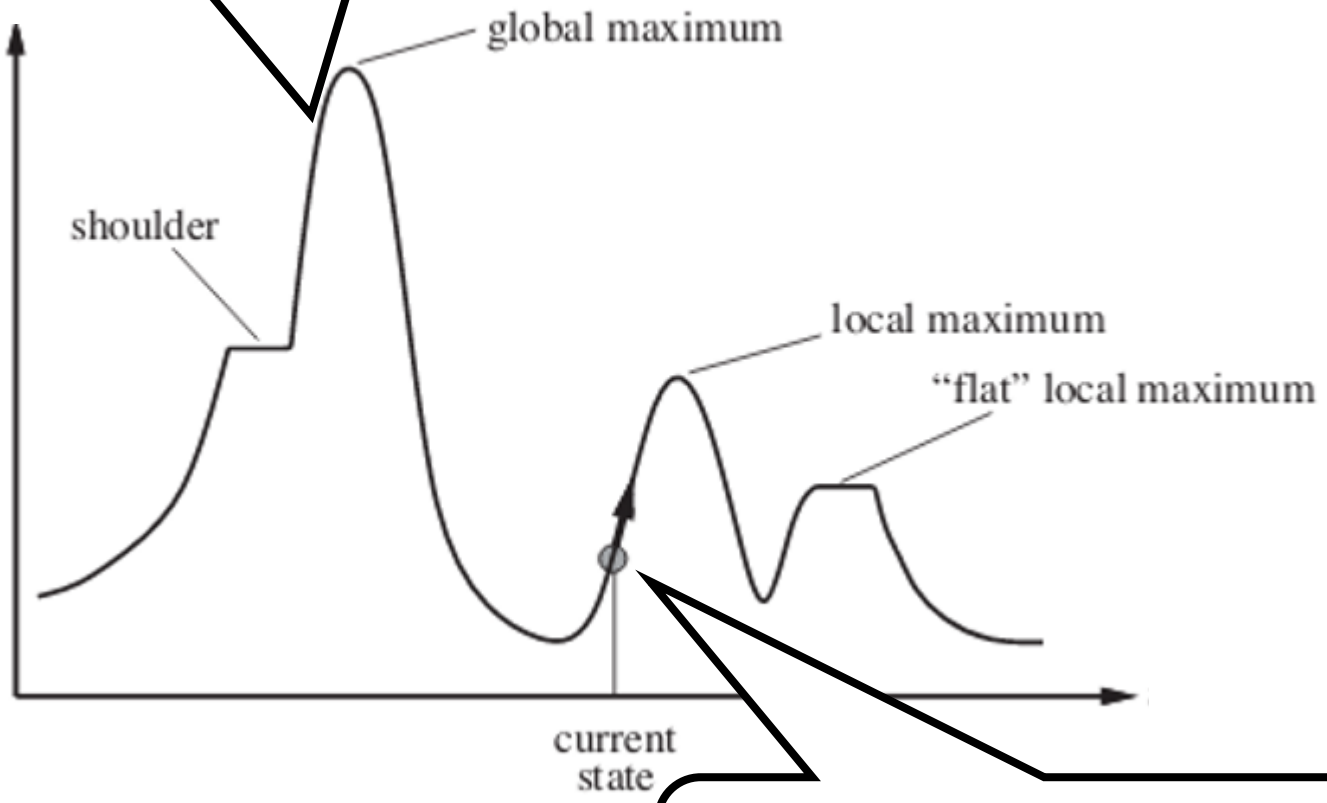
$$3,\ \overline{1} \vee \overline{2},\ 1 \vee \overline{3}$$

starting point

1	2	3	profit
F	F	F	2
F	T	F	2
F	T	T	2

stuck

Starting here will succeed.



You may never get the answer if you start here.

Strawman

RROBLEM

Given a CNF formula, check its satisfiability efficiently

GSAT(random + greedy)

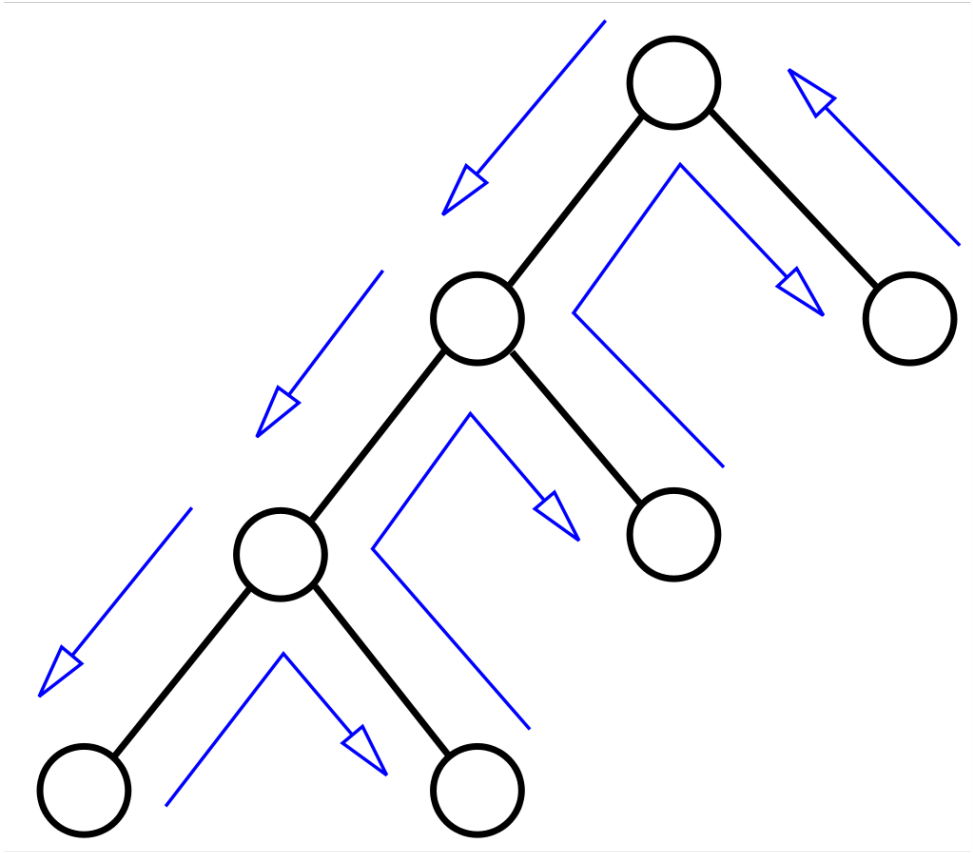
Loop n times

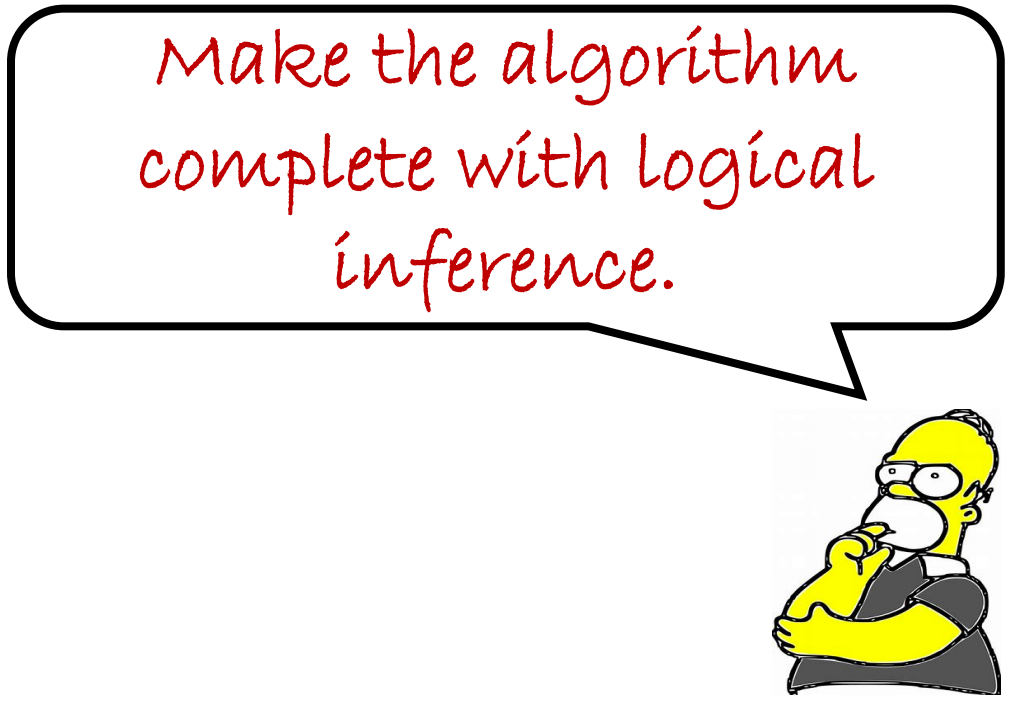
- Randomly select start assignment of variables
- Loop m times
  - Flip a variable that results in maximum profit
  - Exit if profit = number of clauses

ISSUE

The algorithm is not complete. (there's always a chance it will miss an existing solution)

DPLL (Davis-Putnam-Logemann-Loveland Algorithm)





The backtracking DPLL algorithm is the basis for most modern SAT solvers.

DPLL

OVERVIEW

Iteratively process by following steps:

- Guess the truth value of an undefined literal
- Infer the truth value of other variables with inference rules
- If a clause is false and there is a decision literal, backtrack and re-guess a variable

DEFINITION

A literal is defined if its truth value has been guessed or inferred.

DPLL

OVERVIEW

Iteratively process by following steps:

- Guess the truth value of an undefined literal
- Infer the truth value of other variables with inference rules
- If a clause is false and there is a decision literal, backtrack and re-guess a variable

DEFINITION

A literal is defined if its truth value has been guessed or inferred.

DPLL

DECIDE RULE

A literal *l* is defined by ”guessing”, then it is annotated as a “decision literal”. If all defined literals can not satisfy formula F, then ¬*l* must be concerned.

$$(A \vee \neg B) \wedge (B \vee \neg C) \wedge (C \vee A)$$

A is an undefined literal. We can guess A=F temporarily which makes A to be a decision literal.

DPLL

OVERVIEW

Iteratively process by following steps:

- Guess the truth value of an undefined literal
- Infer the truth value of other variables with inference rules
- If a clause is false and there is a decision literal, backtrack and re-guess a variable

DEFINITION

A literal is defined if its truth value has been guessed or inferred.

DPLL

UNITPROPAGRATE RULE

To satisfy a CNF formula, all its clauses have to be true. Hence, if a clause of *F* contains a literal *l* whose truth value is not defined by the current assignment while all the remaining literals of the clause are false, *l* must be defined to be true.

$$(A \vee \neg B) \wedge (B \vee \neg C) \wedge (C \vee A)$$

Because we guess A=F, then we can infer that B=F. Then we can infer C=F according to the second clause.

DPLL

OVERVIEW

Iteratively process by following steps:

- Guess the truth value of an undefined literal
- Infer the truth value of other variables with inference rules
- If a clause is false and there is a decision literal, backtrack and re-guess a variable

DEFINITION

A literal is defined if its truth value has been guessed or inferred.

DPLL

BACKTRACK RULE

If a conflicting clause C is detected and there is defined decision literal *l*, then the rule backtracks by replacing the most recent decision literal *l* by ¬*l* and removing any subsequent literals in the current assignment. Note that ¬*l* is annotated as a non-decision literal, since the other possibility *l* has already been explored.

$$(A \vee \neg B) \wedge (B \vee \neg C) \wedge (C \vee A)$$

A conflicting clause

Because A=B=C=F, then the last clause is F. Now we can backtrack and guess A=T. Now A is not a decision literal.

DPLL

FAIL RULE

This rule detects a conflicting clause C and produces the FailState state whenever there is no decision literals defined.

$$(A \vee \neg B) \wedge (B \vee \neg C) \wedge (C \vee A)$$

A conflicting clause

We can backtrack because A is a decision literal. If there is no decision literal, we can conclude the formula is unsatisfiable.

DPLL

OVERVIEW

Iteratively try to

- Guess the truth value of an undefined literal
- Infer the truth value of other variables with inference rules
- If a clause is false and there is a decision literal, backtrack and reguess a variable

$$(A \vee \neg B) \wedge (B \vee \neg C) \wedge (C \vee A)$$

Now A=T and we can repeat the three steps to get the answer.

DPLL

PURELITERAL RULE

If a literal *l* is pure in *F*, i.e., it occurs in *F* while its negation does not, then *F* is satisfiable only if it defines *l* to be true. Thus, we can define *l* to be true.

$$(A \vee \neg B) \wedge (B \vee \neg C) \wedge (C \vee A)$$

Actually, we don't need to guess A here.
Because A exists and ¬A doesn't exist.
So we can directly determine A=T.

Example

$$\emptyset \parallel \overline{1}\vee\overline{2},\ \ 2\vee 3,\ \ \overline{1}\vee\overline{3}\vee 4,\ \ 2\vee\overline{3}\vee\overline{4},\ \ 1\vee 4 \quad \Longrightarrow \quad \text{(Decide)}$$

Example

$$\emptyset \parallel \overline{1}\vee\overline{2},\ \ 2\vee 3,\ \ \overline{1}\vee\overline{3}\vee 4,\ \ 2\vee\overline{3}\vee\overline{4},\ \ 1\vee 4 \quad \Longrightarrow \quad \text{(Decide)}$$

$$1^{\mathsf{d}} \parallel \overline{1}\vee\overline{2},\ \ 2\vee 3,\ \ \overline{1}\vee\overline{3}\vee 4,\ \ 2\vee\overline{3}\vee\overline{4},\ \ 1\vee 4 \quad \Longrightarrow \quad \text{(UnitPropagate)}$$

Example

$$
\begin{array}{rcl}
\emptyset & \| & \overline{1}\vee\overline{2},\ \ 2\vee 3,\ \ \overline{1}\vee\overline{3}\vee 4,\ \ 2\vee\overline{3}\vee\overline{4},\ \ 1\vee 4 \\
1^{\mathsf{d}} & \| & \overline{1}\vee\overline{2},\ \ 2\vee 3,\ \ \overline{1}\vee\overline{3}\vee 4,\ \ 2\vee\overline{3}\vee\overline{4},\ \ 1\vee 4 \\
1^{\mathsf{d}}\,\overline{2} & \| & \overline{1}\vee\overline{2},\ \ 2\vee 3,\ \ \overline{1}\vee\overline{3}\vee 4,\ \ 2\vee\overline{3}\vee\overline{4},\ \ 1\vee 4
\end{array}
\quad
\begin{array}{l}
\Longrightarrow \quad \text{(Decide)} \\
\Longrightarrow \quad \text{(UnitPropagate)} \\
\Longrightarrow \quad \text{(UnitPropagate)}
\end{array}
$$

Example

$$
\begin{array}{rcl}
\emptyset & \| & \overline{1}\vee\overline{2},\ \ 2\vee 3,\ \ \overline{1}\vee\overline{3}\vee 4,\ \ 2\vee\overline{3}\vee\overline{4},\ \ 1\vee 4 \\
1^{\mathsf{d}} & \| & \overline{1}\vee\overline{2},\ \ 2\vee 3,\ \ \overline{1}\vee\overline{3}\vee 4,\ \ 2\vee\overline{3}\vee\overline{4},\ \ 1\vee 4 \\
1^{\mathsf{d}}\,\overline{2} & \| & \overline{1}\vee\overline{2},\ \ 2\vee 3,\ \ \overline{1}\vee\overline{3}\vee 4,\ \ 2\vee\overline{3}\vee\overline{4},\ \ 1\vee 4 \\
1^{\mathsf{d}}\,\overline{2}\,3 & \| & \overline{1}\vee\overline{2},\ \ 2\vee 3,\ \ \overline{1}\vee\overline{3}\vee 4,\ \ 2\vee\overline{3}\vee\overline{4},\ \ 1\vee 4
\end{array}
\quad
\begin{array}{l}
\Longrightarrow\ \text{(Decide)} \\
\Longrightarrow\ \text{(UnitPropagate)} \\
\Longrightarrow\ \text{(UnitPropagate)} \\
\Longrightarrow\ \text{(UnitPropagate)}
\end{array}
$$

Example

$$
\begin{array}{rcllll}
\emptyset & \| & \overline{1}\vee\overline{2}, & 2\vee 3, & \overline{1}\vee\overline{3}\vee 4, & 2\vee\overline{3}\vee\overline{4}, \quad 1\vee 4 \quad \Longrightarrow & \text{(Decide)} \\
1^{\mathsf{d}} & \| & \overline{1}\vee\overline{2}, & 2\vee 3, & \overline{1}\vee\overline{3}\vee 4, & 2\vee\overline{3}\vee\overline{4}, \quad 1\vee 4 \quad \Longrightarrow & \text{(UnitPropagate)} \\
1^{\mathsf{d}}\,\overline{2} & \| & \overline{1}\vee\overline{2}, & 2\vee 3, & \overline{1}\vee\overline{3}\vee 4, & 2\vee\overline{3}\vee\overline{4}, \quad 1\vee 4 \quad \Longrightarrow & \text{(UnitPropagate)} \\
1^{\mathsf{d}}\,\overline{2}\,3 & \| & \overline{1}\vee\overline{2}, & 2\vee 3, & \overline{1}\vee\overline{3}\vee 4, & 2\vee\overline{3}\vee\overline{4}, \quad 1\vee 4 \quad \Longrightarrow & \text{(UnitPropagate)} \\
1^{\mathsf{d}}\,\overline{2}\,3\,4 & \| & \overline{1}\vee\overline{2}, & 2\vee 3, & \overline{1}\vee\overline{3}\vee 4, & 2\vee\overline{3}\vee\overline{4}, \quad 1\vee 4 \quad \Longrightarrow & \text{(Backtrack)}
\end{array}
$$

Example

$$
\begin{array}{rcllll}
\emptyset & \| & \overline{1}\vee\overline{2}, \quad 2\vee 3, \quad \overline{1}\vee\overline{3}\vee 4, \quad 2\vee\overline{3}\vee\overline{4}, \quad 1\vee 4 & \Longrightarrow & \text{(Decide)} \\
1^{\mathsf{d}} & \| & \overline{1}\vee\overline{2}, \quad 2\vee 3, \quad \overline{1}\vee\overline{3}\vee 4, \quad 2\vee\overline{3}\vee\overline{4}, \quad 1\vee 4 & \Longrightarrow & \text{(UnitPropagate)} \\
1^{\mathsf{d}}\,\overline{2} & \| & \overline{1}\vee\overline{2}, \quad 2\vee 3, \quad \overline{1}\vee\overline{3}\vee 4, \quad 2\vee\overline{3}\vee\overline{4}, \quad 1\vee 4 & \Longrightarrow & \text{(UnitPropagate)} \\
1^{\mathsf{d}}\,\overline{2}\,3 & \| & \overline{1}\vee\overline{2}, \quad 2\vee 3, \quad \overline{1}\vee\overline{3}\vee 4, \quad 2\vee\overline{3}\vee\overline{4}, \quad 1\vee 4 & \Longrightarrow & \text{(UnitPropagate)} \\
1^{\mathsf{d}}\,\overline{2}\,3\,4 & \| & \overline{1}\vee\overline{2}, \quad 2\vee 3, \quad \overline{1}\vee\overline{3}\vee 4, \quad 2\vee\overline{3}\vee\overline{4}, \quad 1\vee 4 & \Longrightarrow & \text{(Backtrack)} \\
\overline{1} & \| & \overline{1}\vee\overline{2}, \quad 2\vee 3, \quad \overline{1}\vee\overline{3}\vee 4, \quad 2\vee\overline{3}\vee\overline{4}, \quad 1\vee 4 & \Longrightarrow & \text{(UnitPropagate)}
\end{array}
$$

Example

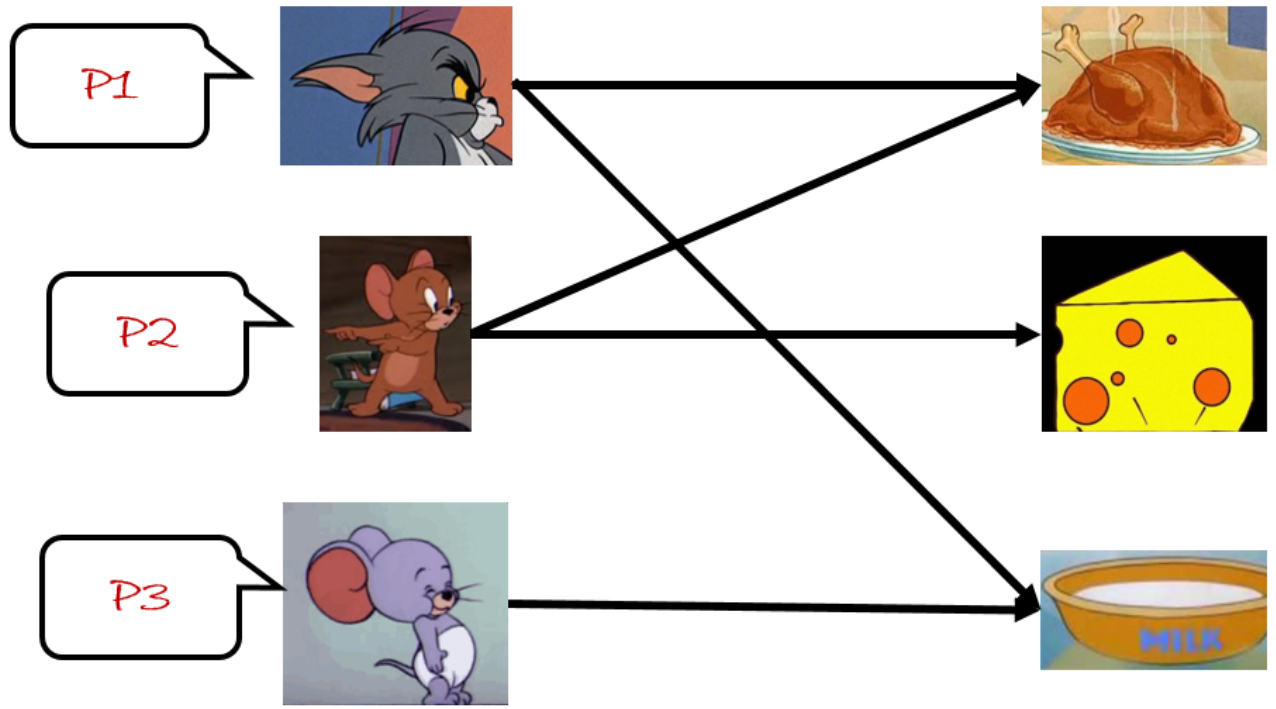
$$
\begin{array}{rcl}
\emptyset & \| & \overline{1}\vee\overline{2},\ \ 2\vee 3,\ \ \overline{1}\vee\overline{3}\vee 4,\ \ 2\vee\overline{3}\vee\overline{4},\ \ 1\vee 4 \quad \Longrightarrow \quad \text{(Decide)} \\
1^{\mathsf{d}} & \| & \overline{1}\vee\overline{2},\ \ 2\vee 3,\ \ \overline{1}\vee\overline{3}\vee 4,\ \ 2\vee\overline{3}\vee\overline{4},\ \ 1\vee 4 \quad \Longrightarrow \quad \text{(UnitPropagate)} \\
1^{\mathsf{d}}\,\overline{2} & \| & \overline{1}\vee\overline{2},\ \ 2\vee 3,\ \ \overline{1}\vee\overline{3}\vee 4,\ \ 2\vee\overline{3}\vee\overline{4},\ \ 1\vee 4 \quad \Longrightarrow \quad \text{(UnitPropagate)} \\
1^{\mathsf{d}}\,\overline{2}\,3 & \| & \overline{1}\vee\overline{2},\ \ 2\vee 3,\ \ \overline{1}\vee\overline{3}\vee 4,\ \ 2\vee\overline{3}\vee\overline{4},\ \ 1\vee 4 \quad \Longrightarrow \quad \text{(UnitPropagate)} \\
1^{\mathsf{d}}\,\overline{2}\,3\,4 & \| & \overline{1}\vee\overline{2},\ \ 2\vee 3,\ \ \overline{1}\vee\overline{3}\vee 4,\ \ 2\vee\overline{3}\vee\overline{4},\ \ 1\vee 4 \quad \Longrightarrow \quad \text{(Backtrack)} \\
\overline{1} & \| & \overline{1}\vee\overline{2},\ \ 2\vee 3,\ \ \overline{1}\vee\overline{3}\vee 4,\ \ 2\vee\overline{3}\vee\overline{4},\ \ 1\vee 4 \quad \Longrightarrow \quad \text{(UnitPropagate)} \\
\overline{1}\,4 & \| & \overline{1}\vee\overline{2},\ \ 2\vee 3,\ \ \overline{1}\vee\overline{3}\vee 4,\ \ 2\vee\overline{3}\vee\overline{4},\ \ 1\vee 4 \quad \Longrightarrow \quad \text{(Decide)}
\end{array}
$$

Example

$$
\begin{array}{rcl}
\emptyset & \| & \overline{1}\vee\overline{2},\ \ 2\vee3,\ \ \overline{1}\vee\overline{3}\vee4,\ \ 2\vee\overline{3}\vee\overline{4},\ \ 1\vee4 \quad \Longrightarrow \quad \text{(Decide)} \\
1^{\mathsf{d}} & \| & \overline{1}\vee\overline{2},\ \ 2\vee3,\ \ \overline{1}\vee\overline{3}\vee4,\ \ 2\vee\overline{3}\vee\overline{4},\ \ 1\vee4 \quad \Longrightarrow \quad \text{(UnitPropagate)} \\
1^{\mathsf{d}}\,\overline{2} & \| & \overline{1}\vee\overline{2},\ \ 2\vee3,\ \ \overline{1}\vee\overline{3}\vee4,\ \ 2\vee\overline{3}\vee\overline{4},\ \ 1\vee4 \quad \Longrightarrow \quad \text{(UnitPropagate)} \\
1^{\mathsf{d}}\,\overline{2}\,3 & \| & \overline{1}\vee\overline{2},\ \ 2\vee3,\ \ \overline{1}\vee\overline{3}\vee4,\ \ 2\vee\overline{3}\vee\overline{4},\ \ 1\vee4 \quad \Longrightarrow \quad \text{(UnitPropagate)} \\
1^{\mathsf{d}}\,\overline{2}\,3\,4 & \| & \overline{1}\vee\overline{2},\ \ 2\vee3,\ \ \overline{1}\vee\overline{3}\vee4,\ \ 2\vee\overline{3}\vee\overline{4},\ \ 1\vee4 \quad \Longrightarrow \quad \text{(Backtrack)} \\
\overline{1} & \| & \overline{1}\vee\overline{2},\ \ 2\vee3,\ \ \overline{1}\vee\overline{3}\vee4,\ \ 2\vee\overline{3}\vee\overline{4},\ \ 1\vee4 \quad \Longrightarrow \quad \text{(UnitPropagate)} \\
\overline{1}\,4 & \| & \overline{1}\vee\overline{2},\ \ 2\vee3,\ \ \overline{1}\vee\overline{3}\vee4,\ \ 2\vee\overline{3}\vee\overline{4},\ \ 1\vee4 \quad \Longrightarrow \quad \text{(Decide)} \\
\overline{1}\,4\,\overline{3}^{\mathsf{d}} & \| & \overline{1}\vee\overline{2},\ \ 2\vee3,\ \ \overline{1}\vee\overline{3}\vee4,\ \ 2\vee\overline{3}\vee\overline{4},\ \ 1\vee4 \quad \Longrightarrow \quad \text{(UnitPropagate)}
\end{array}
$$

Example

$$
\begin{array}{rcl}
\emptyset & \| & \overline{1}\vee\overline{2},\ \ 2\vee 3,\ \ \overline{1}\vee\overline{3}\vee 4,\ \ 2\vee\overline{3}\vee\overline{4},\ \ 1\vee 4 \quad \Longrightarrow \quad \text{(Decide)} \\
1^{\mathsf{d}} & \| & \overline{1}\vee\overline{2},\ \ 2\vee 3,\ \ \overline{1}\vee\overline{3}\vee 4,\ \ 2\vee\overline{3}\vee\overline{4},\ \ 1\vee 4 \quad \Longrightarrow \quad \text{(UnitPropagate)} \\
1^{\mathsf{d}}\,\overline{2} & \| & \overline{1}\vee\overline{2},\ \ 2\vee 3,\ \ \overline{1}\vee\overline{3}\vee 4,\ \ 2\vee\overline{3}\vee\overline{4},\ \ 1\vee 4 \quad \Longrightarrow \quad \text{(UnitPropagate)} \\
1^{\mathsf{d}}\,\overline{2}\,3 & \| & \overline{1}\vee\overline{2},\ \ 2\vee 3,\ \ \overline{1}\vee\overline{3}\vee 4,\ \ 2\vee\overline{3}\vee\overline{4},\ \ 1\vee 4 \quad \Longrightarrow \quad \text{(UnitPropagate)} \\
1^{\mathsf{d}}\,\overline{2}\,3\,4 & \| & \overline{1}\vee\overline{2},\ \ 2\vee 3,\ \ \overline{1}\vee\overline{3}\vee 4,\ \ 2\vee\overline{3}\vee\overline{4},\ \ 1\vee 4 \quad \Longrightarrow \quad \text{(Backtrack)} \\
\overline{1} & \| & \overline{1}\vee\overline{2},\ \ 2\vee 3,\ \ \overline{1}\vee\overline{3}\vee 4,\ \ 2\vee\overline{3}\vee\overline{4},\ \ 1\vee 4 \quad \Longrightarrow \quad \text{(UnitPropagate)} \\
\overline{1}\,4 & \| & \overline{1}\vee\overline{2},\ \ 2\vee 3,\ \ \overline{1}\vee\overline{3}\vee 4,\ \ 2\vee\overline{3}\vee\overline{4},\ \ 1\vee 4 \quad \Longrightarrow \quad \text{(Decide)} \\
\overline{1}\,4\,\overline{3}^{\mathsf{d}} & \| & \overline{1}\vee\overline{2},\ \ 2\vee 3,\ \ \overline{1}\vee\overline{3}\vee 4,\ \ 2\vee\overline{3}\vee\overline{4},\ \ 1\vee 4 \quad \Longrightarrow \quad \text{(UnitPropagate)} \\
\overline{1}\,4\,\overline{3}^{\mathsf{d}}\,2 & \| & \overline{1}\vee\overline{2},\ \ 2\vee 3,\ \ \overline{1}\vee\overline{3}\vee 4,\ \ 2\vee\overline{3}\vee\overline{4},\ \ 1\vee 4
\end{array}
$$



$$\emptyset \parallel 3,\ \overline{1} \vee \overline{2},\ 1 \vee \overline{3} \implies (PureLiteral)$$

$$\overline{2} \parallel 3,\ \overline{1} \vee \overline{2},\ 1 \vee \overline{3} \implies (UnitProp)$$

$$\overline{2}\,3 \parallel 3,\ \overline{1} \vee \overline{2},\ 1 \vee \overline{3} \implies (UnitProp)$$

$$\overline{2}\,3\,1 \parallel 3,\ \overline{1} \vee \overline{2},\ 1 \vee \overline{3}$$

Exercise

$$1 \vee \overline{2}, \quad \overline{1} \vee \overline{2}, \quad 2 \vee 3, \quad \overline{3} \vee 2, \quad 1 \vee 4$$