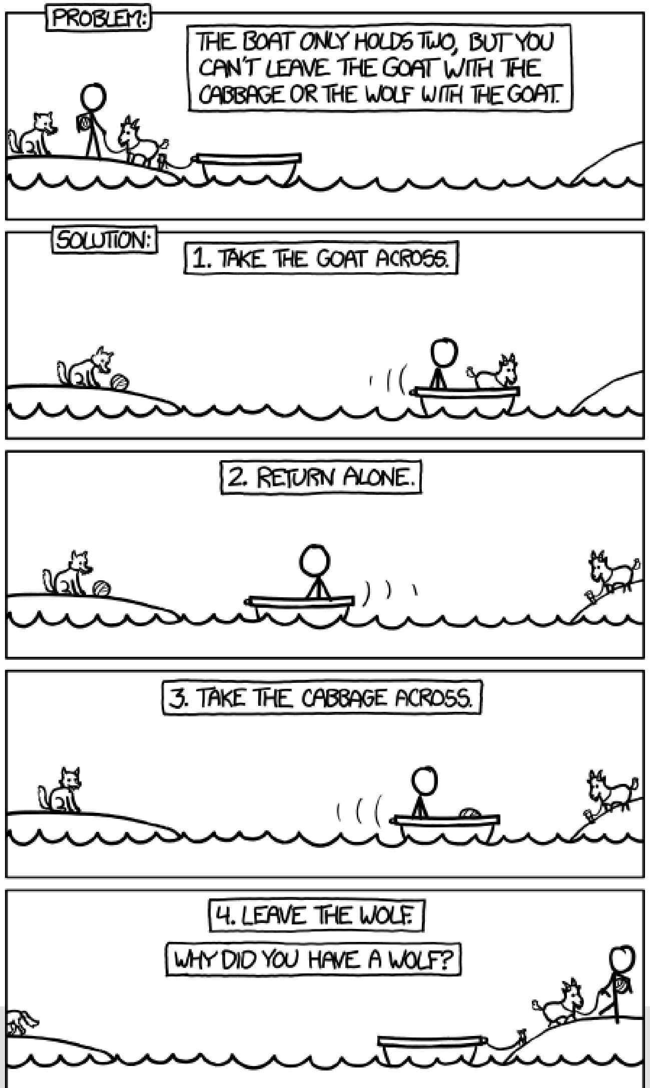
Logic and Computability

Lecture 4

SAT Solver

Bettina Könighofer
bettina.koenighofer@lamarr.at

Stefan Pranger
stefan.pranger@iaik.tugraz.at



https://xkcd.com/287/

Motivation

- Applications
  - HW and SW Verification
  - Bounded Model Checking
  - (Hardware) Equivalence Checking
  - Circuit Synthesis
  - Planning (e.g., air-traffic control, telegraph routing)
  - Scheduling (sport tournaments)
  - Finite mathematics
  - Cryptanalysis
  - …

Motivation – SAT Encoding

- Automatically generated from problem specifications

```
p cnf 51639 368352
−1 7 0
−1 6 0
−1 5 0
−1 −4 0
−1 3 0
−1 2 0
−1 −8 0
−9 15 0
−9 14 0
−9 13 0
−9 −12 0
−9 11 0
−9 10 0
−9 −16 0
−17 23 0
−17 22 0
```

$\neg x_1 \lor x_7$

$\neg x_1 \lor x_6$

$\neg x_1 \lor x_5$

Should x_1 be set to false?

10 Pages Later

- Automatically generated from problem specifications

185 −9 0
185 −1 0
177 169 161 153 145 137 129 121 113 105 97
89 81 73 65 57 49 41
33 25 17 9 1 −185 0
186 −187 0
186 −188 0
...

i.e., (x_{177} or x_{169} or x_{161} or x_{153} …
x_{33} or x_{25} or x_{17} or x_9 or x_1 or (not x_{185}))

Note x_1

4.000 Pages Later

- Automatically generated from problem specifications

```
10236 −10050 0
10236 −10051 0
10236 −10235 0
10008 10009 10010 10011 10012 10013 10014
 10015 10016 10017 10018 10019 10020 10021
 10022 10023 10024 10025 10026 10027 10028
 10029 10030 10031 10032 10033 10034 10035
 10036 10037 10086 10087 10088 10089 10090
 10091 10092 10093 10094 10095 10096 10097
 10098 10099 10100 10101 10102 10103 10104
 10105 10106 10107 10108 −55 −54 53 −52 −51 50
 10047 10048 10049 10050 10051 10235 −10236 0
10237 −10008 0
10237 −10009 0
10237 −10010 0
```

...

Finally, 15.000 Pages Later

```
-7 260 0
7 -260 0
1072 1070 0
-15 -14 -13 -12 -11 -10 0
-15 -14 -13 -12 -11 10 0
-15 -14 -13 -12 11 -10 0
-15 -14 -13 -12 11 10 0
-7 -6 -5 -4 -3 -2 0
-7 -6 -5 -4 -3 2 0
-7 -6 -5 -4 3 -2 0
-7 -6 -5 -4 3 2 0
185 0
```

- How long to solve it?
  - Modern SAT solver needs just a few seconds!

The SAT Problem

[Lecture] Define the *Boolean Satisfiability Problem*?

The SAT Problem

[Lecture] Define the *Boolean Satisfiability Problem*?

Given a propositional formula φ, the Boolean Satisfiability Problem asks whether there exists a model such that the formula evaluates to true.

The SAT Problem

[Lecture] What is the complexity of the SAT-Problem? What does its complexity imply?

The SAT Problem

[Lecture] What is the complexity of the SAT-Problem? What does its complexity imply?

The SAT-Problem is NP-complete.
Its complexity implies that it is very unlikely that there exists any polynomial algorithm.

Outline

- Normal Forms
- DPLL Algorithm
  - Boolean Constrain Propagation
  - Pure Literals
- Conflict-Driven Clause Learning
- Resolution Proofs
  - Resolution Rule
- Tseitin's Algorithm (if time allows it)

Terminology

- **Literal:** propositional variable or its negation
  - Example: p, q, r
- **Clause:** disjunction of literals
  - Example: $(p \vee q \vee \neg r)$
- **Cube:** conjunction of literals
  - Example: $(q \wedge \neg q \wedge \neg r)$

Normal Forms

- **Disjunctive Normal Form (DNF)**
  - Disjunction of cubes:

$$(a_1 \wedge a_2 \wedge \ldots \wedge a_n) \vee (b_1 \wedge b_2 \wedge \ldots \wedge b_m) \vee \ldots$$

  where each a_i, b_j is a literal.

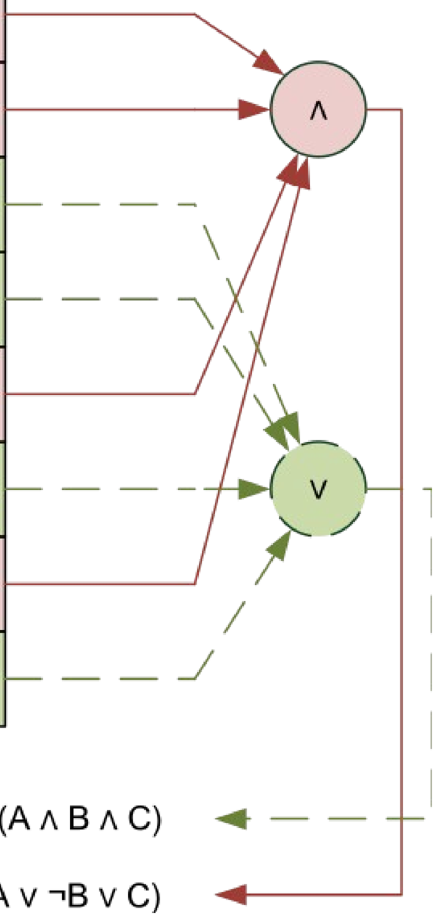
- **Conjunctive Normal Form (CNF)**
  - Conjunction of clauses:

$$(a_1 \vee a_2 \vee \ldots \vee a_n) \wedge (b_1 \vee b_2 \vee \ldots \vee b_m) \vee \ldots$$

  where each a_i, b_j is a literal.

Normal Forms

A	B	C	Ergebnis	Klausel
0	0	0	0	A ∨ B ∨ C
0	0	1	0	A ∨ B ∨ ¬C
0	1	0	1	¬A ∧ B ∧ ¬C
0	1	1	1	¬A ∧ B ∧ C
1	0	0	0	¬A ∨ B ∨ C
1	0	1	1	A ∧ ¬B ∧ C
1	1	0	0	¬A ∨ ¬B ∨ C
1	1	1	1	A ∧ B ∧ C



DNF: (¬A ∧ B ∧ ¬C) ∨ (¬A ∧ B ∧ C) ∨ (A ∧ ¬B ∧ C) ∨ (A ∧ B ∧ C)

KNF: (A ∨ B ∨ C) ∧ (A ∨ B ∨ ¬C) ∧ (¬A ∨ B ∨ C) ∧ (¬A ∨ ¬B ∨ C)

https://de.wikipedia.org/wiki/Konjunktive_Normalform

Normal Forms

[Lecture] Given the formula $\varphi = (q \rightarrow p) \wedge (r \vee \neg p)$. Compute its representation in Disjunctive Normal Form (DNF) using a truth table.

[Lecture] Given the formula $\varphi = (q \rightarrow p) \wedge (r \vee \neg p)$. Compute its representation in Conjunctive Normal Form (CNF) using a truth table.

p	q	r	$\neg p$	$r \vee \neg p$	$q \rightarrow p$	φ
F	**F**	**F**	**T**	**T**	**T**	**T**
F	**F**	**T**	**T**	**T**	**T**	**T**
F	**T**	**F**	**T**	**T**	**F**	**F**
F	**T**	**T**	**T**	**T**	**F**	**F**
T	**F**	**F**	**F**	**F**	**T**	**F**
T	**F**	**T**	**F**	**T**	**T**	**T**
T	**T**	**F**	**F**	**F**	**T**	**F**
T	**T**	**T**	**F**	**T**	**T**	**T**

Normal Forms

[Lecture] Given the formula $\varphi = (q \rightarrow p) \wedge (r \vee \neg p)$. Compute its representation in Disjunctive Normal Form (DNF) using a truth table.

[Lecture] Given the formula $\varphi = (q \rightarrow p) \wedge (r \vee \neg p)$. Compute its representation in Conjunctive Normal Form (CNF) using a truth table.

p	q	r	$\neg p$	$r \vee \neg p$	$q \rightarrow p$	φ
F	F	F	T	T	T	T
F	F	T	T	T	T	T
F	T	F	T	T	F	F
F	T	T	T	T	F	F
T	F	F	F	F	T	F
T	F	T	F	T	T	T
T	T	F	F	F	T	F
T	T	T	F	T	T	T

$$DNF(\varphi) = (\neg p \wedge \neg q \wedge \neg r) \vee (\neg p \wedge \neg q \wedge r) \vee (p \wedge \neg q \wedge r) \vee (p \wedge q \wedge r)$$

Normal Forms

[Lecture] Given the formula $\varphi = (q \to p) \wedge (r \vee \neg p)$. Compute its representation in Disjunctive Normal Form (DNF) using a truth table.

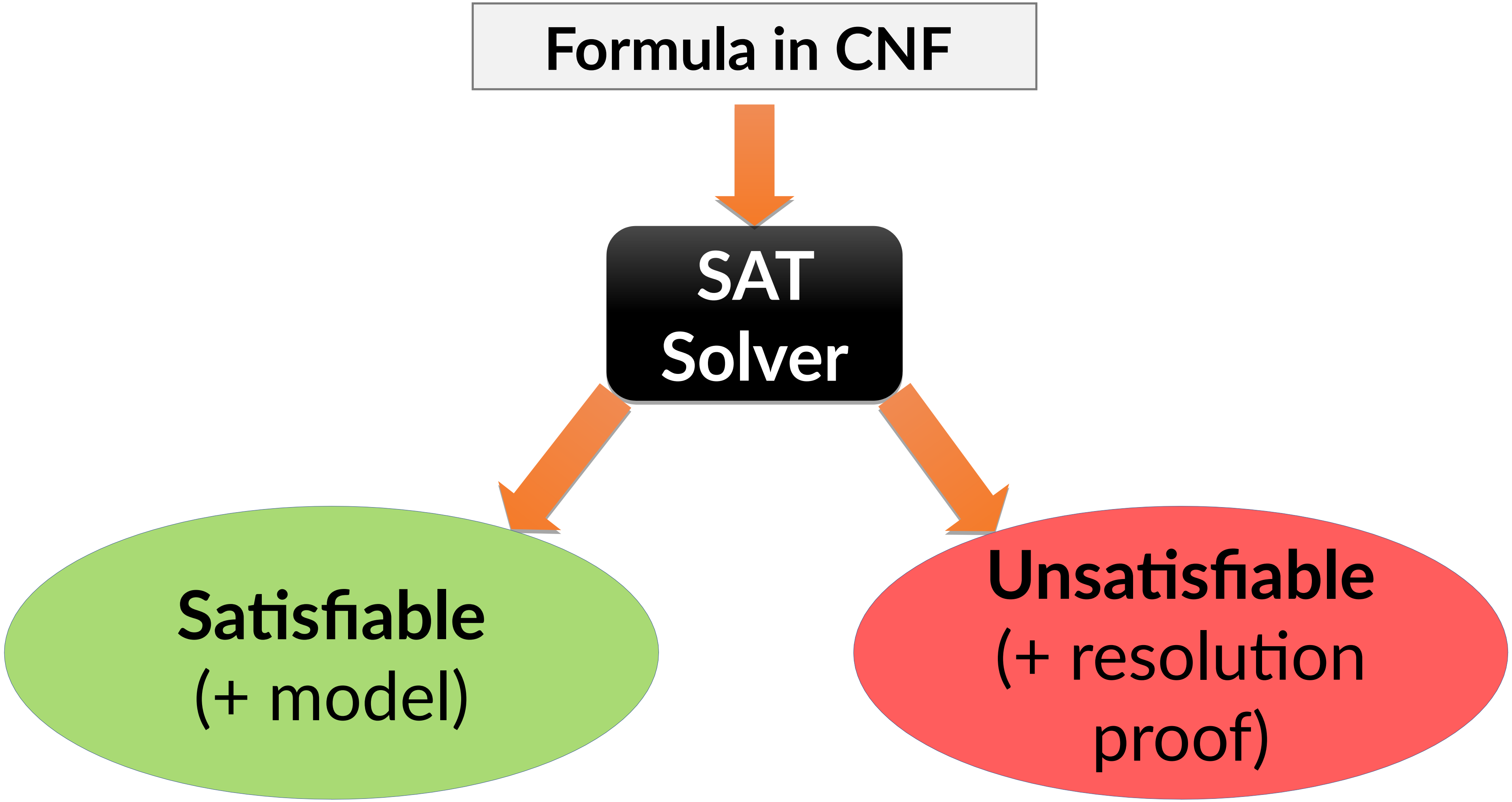
[Lecture] Given the formula $\varphi = (q \to p) \wedge (r \vee \neg p)$. Compute its representation in Conjunctive Normal Form (CNF) using a truth table.

p	q	r	$\neg p$	$r \vee \neg p$	$q \to p$	φ
F	**F**	**F**	**T**	**T**	**T**	**T**
F	**F**	**T**	**T**	**T**	**T**	**T**
F	**T**	**F**	**T**	**T**	**F**	**F**
F	**T**	**T**	**T**	**T**	**F**	**F**
T	**F**	**F**	**F**	**F**	**T**	**F**
T	**F**	**T**	**F**	**T**	**T**	**T**
T	**T**	**F**	**F**	**F**	**T**	**F**
T	**T**	**T**	**F**	**T**	**T**	**T**

$$
\begin{aligned}
DNF(\varphi) = & (\neg p \wedge \neg q \wedge \neg r) \\
& \vee (\neg p \wedge \neg q \wedge r) \\
& \vee (p \wedge \neg q \wedge r) \\
& \vee (p \wedge q \wedge r)
\end{aligned}
$$

$$
\begin{aligned}
CNF(\varphi) = & (p \vee \neg q \vee r) \\
& \wedge (p \vee \neg q \vee \neg r) \\
& \wedge (\neg p \vee q \vee r) \\
& \wedge (\neg p \vee \neg q \vee r)
\end{aligned}
$$

SAT-Solver


Formula in CNF
SAT
Solver
Satisfiable
(+ model)
Unsatisfiable
(+ resolution
proof)

DPLL Algorithm

- Due to **Davis, Putnam, Loveland, Logemann**
    - M. Davis, H. Putnam. "*A computing procedure for quantification theory*". Journal of the ACM, 7:201-215, 1960
    - M. Davis, G. Logemann, and D. Loveland. "*A machine program for theorem-proving*". Communications of the ACM, 5:394-397, 1962
- Basis for most modern SAT solvers

Notation

- φ: formula in CNF
  - E.g., $\varphi = (a \vee b \vee \neg d) \wedge c$
- **A**: Assignment
  - given in set representation, e.g.: $\{\neg a, b, d\}$
  - conjunction of literals, e.g. $A = \neg a \wedge b \wedge d$
  - Total or partial Assignment
- **φ[A]**: φ with variables set according to A
  - E.g., $\varphi[A] =$

Notation

- φ: formula in CNF
  - E.g., $\varphi = (a \lor b \lor \neg d) \land c$
- **A**: Assignment
  - given in set representation, e.g.: $\{\neg a, b, d\}$
  - conjunction of literals, e.g. $A = \neg a \land b \land d$
  - Total or partial Assignment
- **φ[A]**: φ with variables set according to A
  - E.g., $\varphi[A] = (FALSE \land TRUE \land \neg TRUE) \land c = c$

Basis Idea - Backtracking Binary Search

- Recursively search an A:
  - φ[A] is TRUE
    - Proves φ satisfiable
    - "A" is a satisfying model
- No such A exists
  - φ is unsatisfiable

CNF is a Set of Clauses

- Formula:
  - $\varphi =$
    $(a \lor \neg b \lor c) \land$
    $(\neg a \lor \neg d) \land$
    $(\neg c)$

- Set Representation:
  - $C = \{$
    $\{a, \neg b, c\},$
    $\{\neg a, \neg d\},$
    $\{\neg c\}$
    $\}$

Setting Literals

- Compute $\varphi[l]$, for a literal l:
  - Remove all clauses that contain l:
    - They are true
    - E.g. $\varphi = (a \vee b) \wedge c$, $A = \{a\} \rightarrow \varphi[A] = (\text{TRUE} \vee b) \wedge c = (\text{TRUE}) \wedge c$
  - Remove literals $\neg l$ from clauses that contain $\neg l$:
    - They cannot be set to true anymore
    - E.g. $\varphi = (a \vee b) \wedge c$, $A = \{\neg a\} \rightarrow \varphi[A] = (\text{FALSE} \vee b) \wedge c = b \wedge c$

- Truth Value of a CNF
  - An **empty clause** is false $(\text{FALSE} \vee \text{FALSE} \vee \dots) \wedge \dots.$
  - An **<u>set</u> of 'satisfied' clauses** is true $(\text{TRUE}) \wedge (\text{TRUE}) \wedge \dots$

DPLL Example

[Lecture] Use the DPLL algorithm (*without* BCP, PL and clause learning) to determine whether or not the set of clauses given is satisfiable. Decide variables in alphabetical order starting with the *positive* phase. If the set of clauses resulted in SAT, give a satisfying model.

Clause 1: $(\neg a \vee b)$

Clause 2: $(\neg b \vee c)$

Clause 3: $(\neg c \vee d)$

Clause 4: $(\neg d \vee e)$

Clause 5: $(\neg e \vee \neg a)$

DPLL Example

Step	1	2	3	4	5	6	7	8	9	10
Decision Level	0	1	2	3	4	5	5	4	3	2
Assignment	-	a	a, b	a, b, c	a, b, c, d	a, b, c, d, e	$a, b, c, d, \neg e$	$a, b, c, \neg d$	$a, b, \neg c$	$a, \neg b$
Cl. 1: $\neg a, b$	1	b	✓	✓	✓	✓	✓	✓	✓	{} ✗
Cl. 2: $\neg b, c$	2	2	c	✓	✓	✓	✓	✓	{} ✗	✓
Cl. 3: $\neg c, d$	3	3	3	d	✓	✓	✓	{} ✗	✓	3
Cl. 4: $\neg d, e$	4	4	4	4	e	✓	{} ✗	✓	4	4
Cl. 5: $\neg e, \neg a$	5	$\neg e$	$\neg e$	$\neg e$	$\neg e$	{} ✗	✓	$\neg e$	$\neg e$	$\neg e$
Decision	a	b	c	d	e	$\neg e$	$\neg d$	$\neg c$	$\neg b$	$\neg a$

Step	11	12	13	14	15
Decision Level	1	2	3	4	5
Assignment	$\neg a$	$\neg a, b$	$\neg a, b, c$	$\neg a, b, c, d$	$\neg a, b, c, d, e$
Cl. 1: $\neg a, b$	✓	✓	✓	✓	✓
Cl. 2: $\neg b, c$	2	c	✓	✓	✓
Cl. 3: $\neg c, d$	3	3	d	✓	✓
Cl. 4: $\neg d, e$	4	4	4	e	✓
Cl. 5: $\neg e, \neg a$	✓	✓	✓	✓	✓
Decision	b	c	d	e	SAT

Model:

$a = \mathrm{F},\ b = \mathrm{T},\ c = \mathrm{T},\ d = \mathrm{T},\ e = \mathrm{T}$

Decision Heuristic

- Which literal to pick?
  - Randomly
  - According to some order
  - Satisfies largest number of unsatisfied clauses
    - satisfy a clause = occur in a clause

- Open Research Topic

Unit Clauses

- Unit clause:
  - a clause with a single unassigned literal
  - Examples:
    - {a}
    - {¬b}

Unit Clauses

[Lecture] In the context of the DPLL algorithm, explain what a *Unit Clause* is. Give an example.

Unit Clauses

[Lecture] In the context of the DPLL algorithm, explain what a *Unit Clause* is. Give an example.

Definition - Unit Clause. A clause c is said to be a unit clause under some assignment A if the following two conditions hold:

(a) The clause c is not satisfied by A.

(b) All but one of the variables in c are given a value by A.

Therefore, there is a single literal left in the set representing the clause under the assignment.
An example would be:

- $c = \{a, b, \neg c\}$
- $A = \{\neg a, c\}$
- $c[A] = \bot \vee b \vee \bot$, in set representation: $\{c\}$

Boolean Constrain Propagation (BCP)

- Unit clause:
  - a clause with a single unassigned literal
  - Examples:
    - {a}
    - {¬b}

- Unit Clause exists → set its literal
  - Otherwise: immediately FALSE
  - Very simple but very important heuristic!

DPLL + BCP Example

[Lecture] Use the DPLL algorithm (*without* BCP, PL and clause learning) to determine whether or not the set of clauses given is satisfiable. Decide variables in alphabetical order starting with the *positive* phase. If the set of clauses resulted in SAT, give a satisfying model.

Clause 1: $(\neg a \vee b)$

Clause 2: $(\neg b \vee c)$

Clause 3: $(\neg c \vee d)$

Clause 4: $(\neg d \vee e)$

Clause 5: $(\neg e \vee \neg a)$

DPLL + BCP Example

DPLL algorithm:

Step	1	2	3	4	5	6	7	8	9	10	11
Decision Level	0	1	1	1	1	1	1	2	2	2	2
Assignment	-	a	a, b	a, b, c	a, b, c, d	a, b, c, d, e	$\neg a$	$\neg a, b$	$\neg a, b, c$	$\neg a, b, c, d$	$\neg a, b, c, d, e$
Cl. 1: $\neg a, b$	1	b	✓	✓	✓	✓	✓	✓	✓	✓	✓
Cl. 2: $\neg b, c$	2	2	c	✓	✓	✓	2	c	✓	✓	✓
Cl. 3: $\neg c, d$	3	3	3	d	✓	✓	3	3	d	✓	✓
Cl. 4: $\neg d, e$	4	4	4	4	e	✓	4	4	4	e	✓
Cl. 5: $\neg e, \neg a$	5	$\neg e$	$\neg e$	$\neg e$	$\neg e$	{} ✗	✓	✓	✓	✓	✓
BCP	-	b	c	d	e	-	-	c	d	e	✓
Decision	a	-	-	-	-	$\neg a$	b	-	-	-	SAT

Model:

$a = \mathrm{F},\ b = \mathrm{T},\ c = \mathrm{T},\ d = \mathrm{T},\ e = \mathrm{T}$

Pure Literals

- **Pure Literal:**
  - Unassigned literal
  - Complement does not occur in any unsatisfied clause

- Pure literals → set to **TRUE**

DPLL + BCP + Pure Literals Example

[Lecture] Use the DPLL algorithm with *Boolean Constrain Propagation* and *Pure Literals* (*without* clause learning) to determine whether or not the set of clauses given is satisfiable. Decide variables in alphabetical order starting with the *positive* phase. If the set of clauses resulted in SAT, give a satisfying model.

Clause 1: $(\neg a \vee b)$

Clause 2: $(\neg b \vee c)$

Clause 3: $(\neg c \vee d)$

Clause 4: $(\neg d \vee e)$

Clause 5: $(\neg e \vee \neg a)$

DPLL + BCP + Pure Literals Example

DPLL algorithm:

Step	1	2	3	4	5
Decision Level	0	0	0	0	0
Assignment	-	$\neg a$	$\neg a, \neg b$	$\neg a, \neg b, \neg c$	$\neg a, \neg b, \neg c, \neg d$
Cl. 1: $\neg a, b$	1	✓	✓	✓	✓
Cl. 2: $\neg b, c$	2	2	✓	✓	✓
Cl. 3: $\neg c, d$	3	3	3	✓	✓
Cl. 4: $\neg d, e$	4	4	4	4	✓
Cl. 5: $\neg e, \neg a$	5	✓	✓	✓	✓
BCP	-	-	-	-	-
PL	$\neg a$	$\neg b$	$\neg c$	$\neg d$	-
Decision	-	-	-	-	SAT

Model:

$a = \text{F}$, $b = \text{F}$, $c = \text{F}$, $d = \text{F}$, $e = \text{F}$

DPLL Heuristics

[Lecture] In the context of the DPLL algorithm, explain why it is advantageous to apply *Boolean Constrain Propagation (BCP)* and *Pure Literals (PL)* before making a decision.

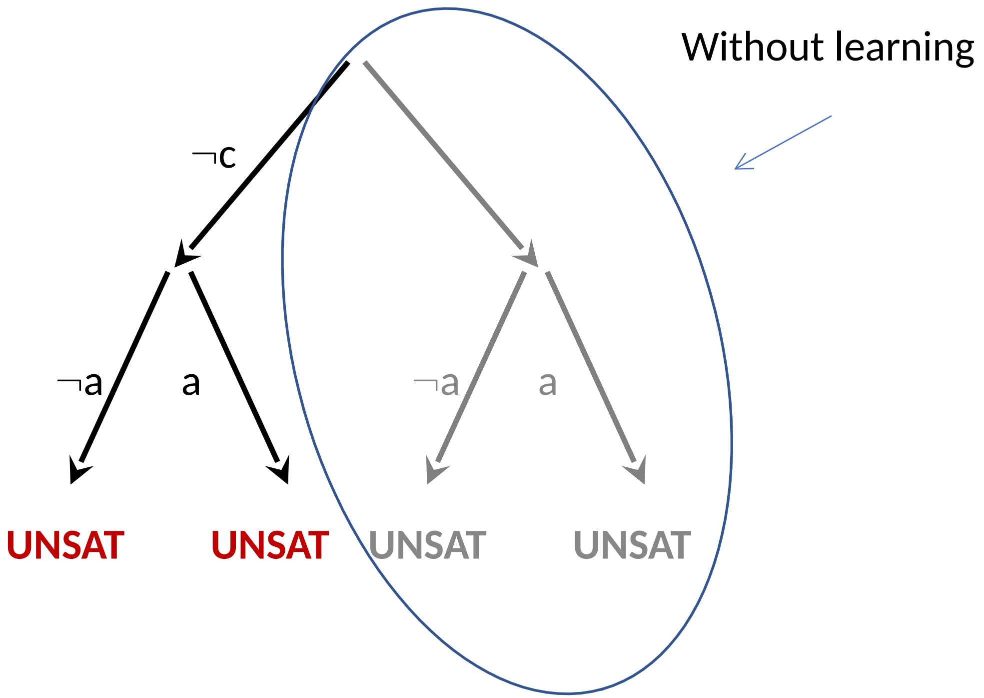
DPLL Heuristics

[Lecture] In the context of the DPLL algorithm, explain why it is advantageous to apply *Boolean Constrain Propagation (BCP)* and *Pure Literals (PL)* before making a decision.

Boolean Constraint Propagation and Pure Literals are so-called heuristics. BCP and PL capture when the choices we can make are restricted in two different ways. It is advantageous to apply these heuristics before making a decision, since it reduces the amount of different assignments we have to check.

Clause Learning

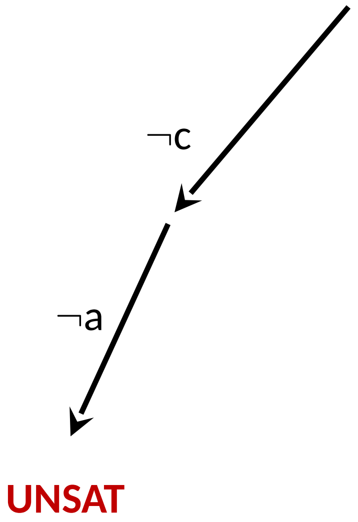
1. (a ∨ ¬c)
2. (b ∨ ¬c)
3. (¬a ∨ ¬b ∨ c)
4. (¬a ∨ ¬b)
5. (¬a ∨ b)
6. (a ∨ ¬b)
7. (a ∨ b)

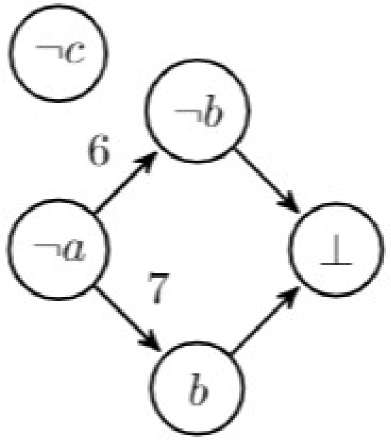


Problem is with the literal "a":
→ No need to try c=TRUE!

Clause Learning

1. $(a \lor \neg c)$
2. $(b \lor \neg c)$
3. $(\neg a \lor \neg b \lor c)$
4. $(\neg a \lor \neg b)$
5. $(\neg a \lor b)$
6. $(a \lor \neg b)$
7. $(a \lor b)$





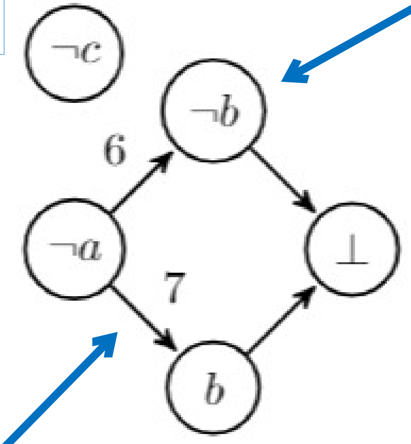
→ **Learn New Clause: (a)**

Conflict Graphs

1. $(a \lor \neg c)$
2. $(b \lor \neg c)$
3. $(\neg a \lor \neg b \lor c)$
4. $(\neg a \lor \neg b)$
5. $(\neg a \lor b)$
6. $(a \lor \neg b)$
7. $(a \lor b)$

Root nodes:
Decisions

Internal nodes:
Implied (by BCP)



Clause numbers:
b is implied by
¬a via clause 7

Conflict Node

Conflict Driven Clause Learning

[Lecture] In the context of the DPLL algorithm, explain what *Conflict-Driven Clause Learning* is and why most modern SAT solvers use this technique.

Conflict Driven Clause Learning

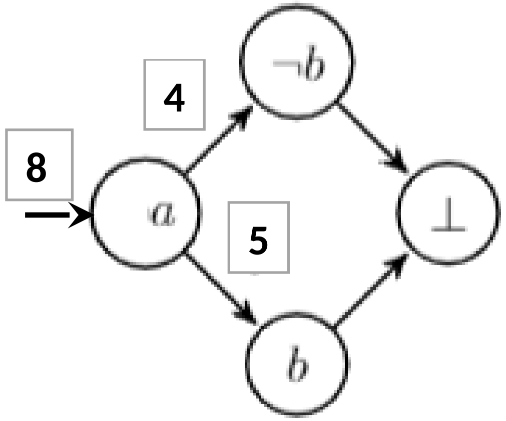
[Lecture] In the context of the DPLL algorithm, explain what *Conflict-Driven Clause Learning* is and why most modern SAT solvers use this technique.

The idea of conflict-driven clause learning is not to repeat steps that lead to a conflict.

When executing the DPLL algorithm we can maintain a so-called conflict graph. We can use this graph to deduce which variables lead to a conflict. In Conflict-Driven Clause Learning different SAT solvers apply different techniques to extract new *learned* clauses from this graph.
The learned clauses help the SAT solver to no repeat mistakes in different execution branches.

Conflict Graphs

1. $(a \lor \neg c)$
2. $(b \lor \neg c)$
3. $(\neg a \lor \neg b \lor c)$
4. $(\neg a \lor \neg b)$
5. $(\neg a \lor b)$
6. $(a \lor \neg b)$
7. $(a \lor b)$



No decision was necessary
→ We learn: UNSAT

Backtrack Level

- Ongoing Research Problem

- In this course:
  - $\rightarrow$ earliest level where conflict clause is a unit clause
  - New clause can immediately be used

DPLL + BCP + PL + CDCL

[Lecture] Use the DPLL algorithm with conflict-driven clause learning to determine whether or not the set of clauses given is satisfiable. Decide variables in alphabetical order starting with the *negative* phase. For conflicts, draw conflict graphs after the end of the table, and add the learned clause to the table.
If the set of clauses resulted in `SAT`, give a satisfying model. If the set of clauses resulted in `UNSAT`, give a resolution proof that shows that the conjunction of the clauses from the table is unsatisfiable.

Clause 1: $\{\neg a, \neg b\}$

Clause 2: $\{a, c\}$

Clause 3: $\{b, \neg c\}$

Clause 4: $\{\neg b, d\}$

Clause 5: $\{\neg c, \neg d\}$

Clause 6: $\{c, e\}$

Clause 7: $\{c, \neg e\}$

DPLL + BCP + PL + CDCL

[Lecture] Use the DPLL algorithm with conflict-driven clause learning to determine whether or not the set of clauses given is satisfiable. Decide variables in alphabetical order starting with the *negative* phase. For conflicts, draw conflict graphs after the end of the table, and add the learned clause to the table.
If the set of clauses resulted in `SAT`, give a satisfying model. If the set of clauses resulted in `UNSAT`, give a resolution proof that shows that the conjunction of the clauses from the table is unsatisfiable.

Clause 1: $\{\neg a, \neg b\}$

Clause 2: $\{a, c\}$

Clause 3: $\{b, \neg c\}$

Clause 4: $\{\neg b, d\}$

Clause 5: $\{\neg c, \neg d\}$

Clause 6: $\{c, e\}$

Clause 7: $\{c, \neg e\}$

https://git.pranger.xyz/sp/LAC-Questionnaire/releases/download/release_three/questionnaire_with_solutions_three.pdf

SAT Solver Output

- **Satisfiable**:
  - Satisfying Assignment

- **Unsatisfiable**
  - Proof of Unsatisfiability

Proving Unsatisfiability

- **Resolution Rule:**

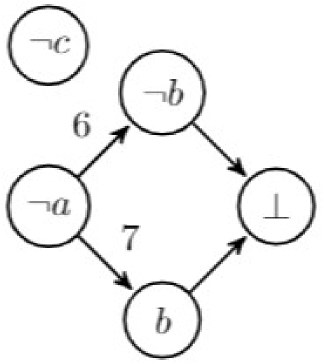
$$\frac{(a \lor b_1 \lor \ldots \lor b_n) \qquad (\neg a \lor c_1 \lor \ldots \lor c_m)}{(b_1 \lor \ldots \lor b_n \lor c_1 \lor \ldots \lor c_m)}$$

- Remember: $\dfrac{a \qquad \neg a}{\text{FALSE}}$

- "Derived" rule for natural deduction

Prove Learned Clause

- Turn Conflict Graph Around
  - Select clause that implies conflict
  - Iteratively, resolve while back-traversing graph

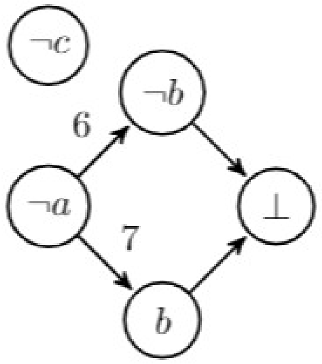


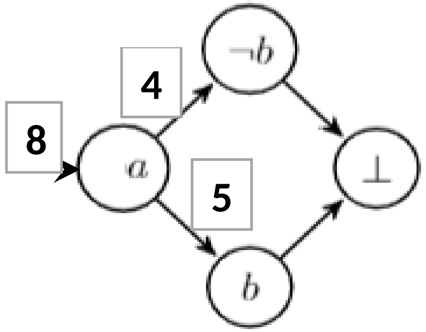
$$\frac{7.\ a \vee b \qquad 6.\ a \vee \neg b}{8.\ a}$$

Cheap Resolution Proof

- Turn All Conflict Graphs Around

1. (a ∨ ¬c)
2. (b ∨ ¬c)
3. (¬a ∨ ¬b ∨ c)
4. (¬a ∨ ¬b)
5. (¬a ∨ b)
6. (a ∨ ¬b)
7. (a ∨ b)



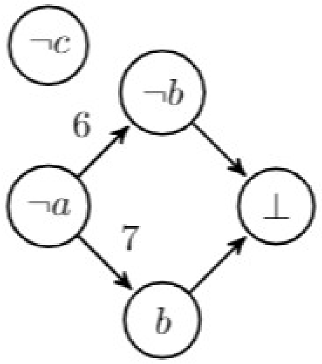


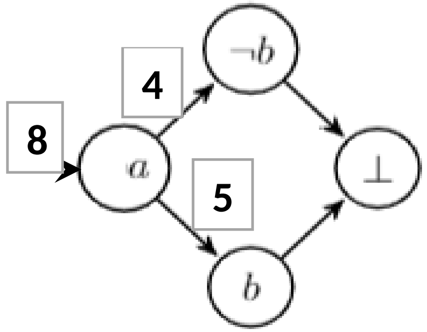
$$\dfrac{\dfrac{7.\ a \lor b \qquad 6.\ a \lor \neg b}{8.\ a} \qquad \dfrac{5.\ \neg a \lor b \qquad 4.\ \neg a \lor \neg b}{\neg a}}{\text{FALSE}}$$

Cheap Resolution Proof

- Turn All Conflict Graphs Around

1. (a ∨ ¬c)
2. (b ∨ ¬c)
3. (¬a ∨ ¬b ∨ c)
4. (¬a ∨ ¬b)
5. (¬a ∨ b)
6. (a ∨ ¬b)
7. (a ∨ b)





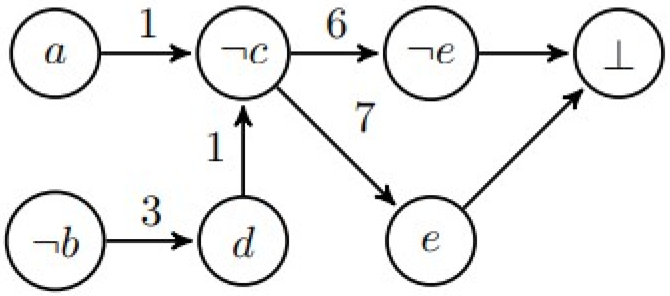
7. a ∨ b   6. a ∨ ¬b   5. ¬a ∨ b   4. ¬a ∨ ¬b

8. a   ¬a

FALSE

Cheap Resolution Proof

[Lecture] Consider the following conflict graph with the following set of clauses:



Clause 1: $\{\neg a, \neg c, \neg d\}$

Clause 2: $\{a, \neg d\}$

Clause 3: $\{b, d\}$

Clause 4: $\{\neg b, d, e\}$

Clause 5: $\{\neg b, \neg e\}$

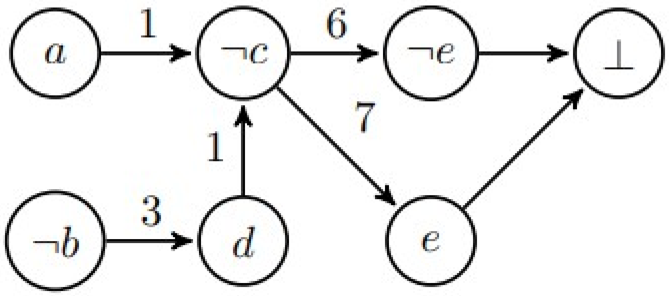
Clause 6: $\{c, \neg e\}$

Clause 7: $\{c, e\}$

Give the resolution proof for the given conflict graph and clauses and state the clause to be learned from the conflict.

Cheap Resolution Proof

[Lecture] Consider the following conflict graph with the following set of clauses:



Clause 1: $\{\neg a, \neg c, \neg d\}$

Clause 2: $\{a, \neg d\}$

Clause 3: $\{b, d\}$

Clause 4: $\{\neg b, d, e\}$

Clause 5: $\{\neg b, \neg e\}$

Clause 6: $\{c, \neg e\}$

Clause 7: $\{c, e\}$

$$\dfrac{\dfrac{\dfrac{6\; c \lor \neg e \qquad 7\; c \lor e}{c} \qquad 1\; \neg a \lor \neg c \lor \neg d}{\neg a \lor \neg d} \qquad 3\; b \lor d}{\neg a \lor b}$$

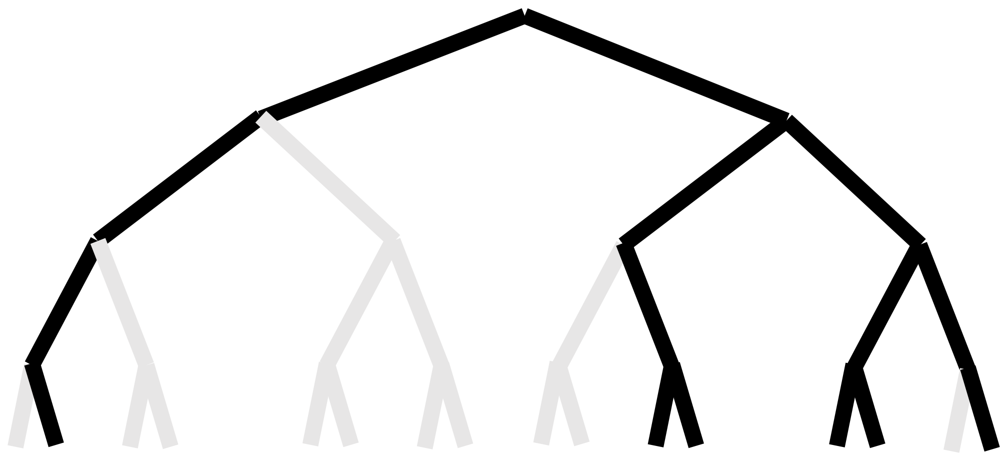
The new learned clause is therefore Cl. 8: $\neg a \lor b$

Give the resolution proof for the given conflict graph and clauses and state the clause to be learned from the conflict.

DPLL + BCP + PL + CDCL

- Binary Search Tree
  - Worst Case: Exponential Time

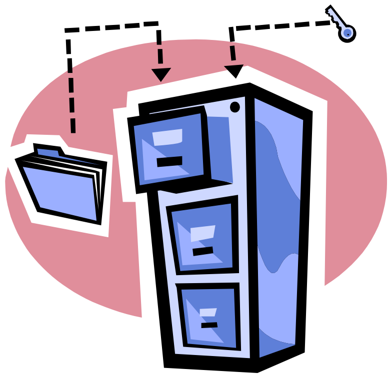
- Pruning
  - Boolean Constraint Propagation (BCP)
  - Pure Literals
  - Learn Conflict Clauses



DPLL - Summary

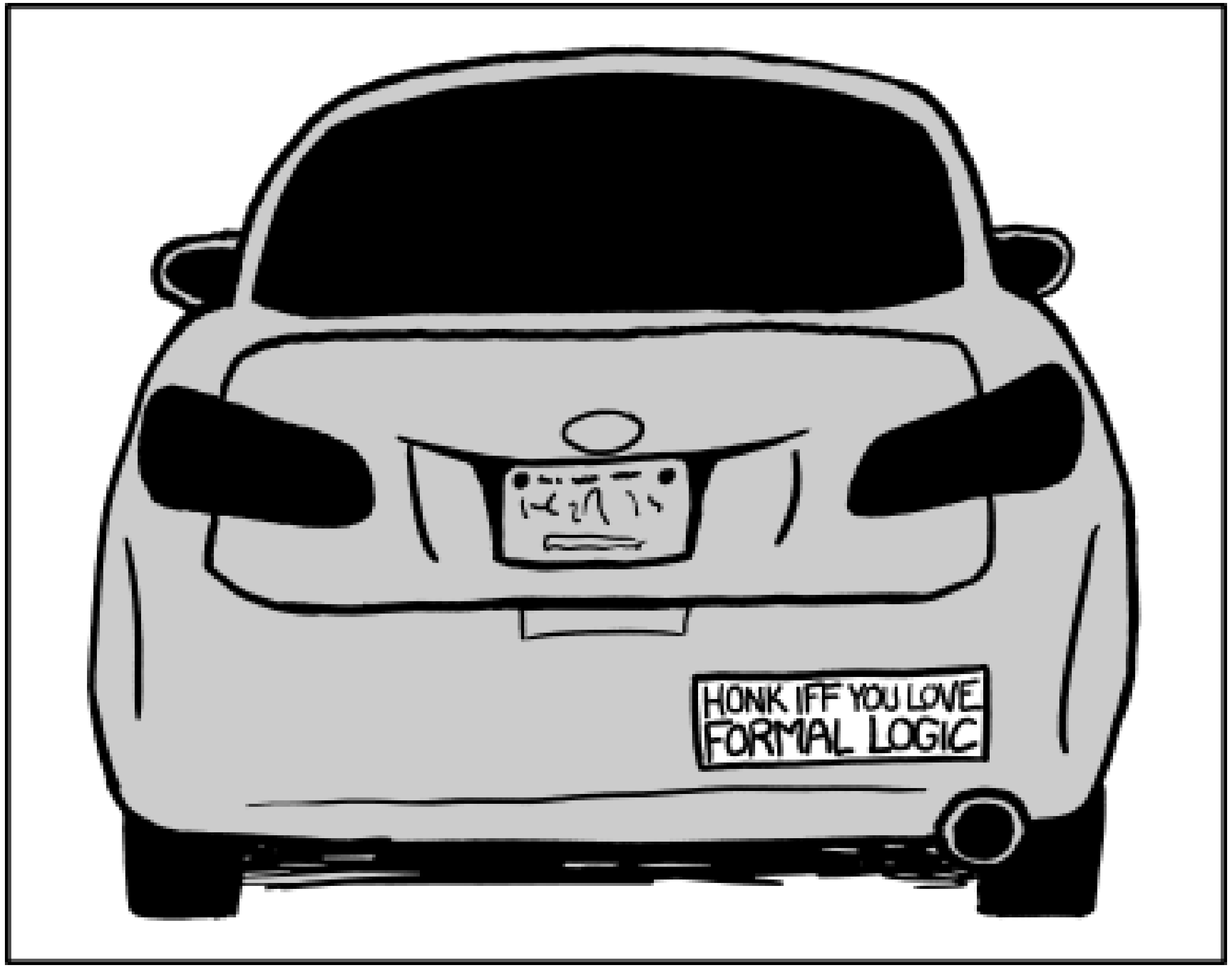
- As long as there is no conflict on decision level 0:
  - Try to perform BCP, if there is no unit clause,
  - try to perform PL, if there is no pure literal,
  - make a decision.
    - Update all clauses.
    - If there is a conflict: Construct graph and resolution proof, add newly learned clause.
    - If all clauses are empty and no conflict: Report satisfying model.
- If there is a conflict on decision level 0:
  - Construct graph and resolution proof

Summary



- DPLL Algorithm
  - Binary Search Tree
  - Worst-Case Exponential
- Pruning
  - Boolean Constraint Propagation
  - Pure Literals
  - Learned Clauses
- Resolution Proofs
  - Resolution Rule
  - “Turn Conflict Graph around”

Thank You



https://xkcd.com/1033/