Logic and Mechanized Reasoning
DP & DPLL

Marijn J.H. Heule

Carnegie Mellon University

Let's First Revisit Resolution

LAMR/Examples/using_sat_solvers/resolution.lean

```
def example0 : Proof := #[
  .hyp clause!{-p -q r}, -- 0
  .hyp clause!{-r},      -- 1
  .hyp clause!{p -q},    -- 2
  .hyp clause!{-s q},    -- 3
  .hyp clause!{s},       -- 4
  .res "r" 0 1,          -- 5 -p -q
  .res "s" 4 3,          -- 6 q
  .res "q" 6 2,          -- 7 p
  .res "p" 7 5,          -- 8 -q
  .res "q" 6 8           -- 9 ⊥
]
```

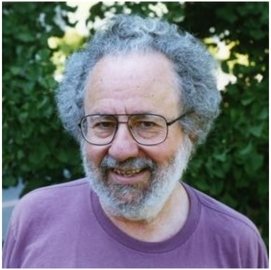
DP Resolution

DPLL

Martin Davis (March 8, 1928 – January 1, 2023)

Martin Davis & Hilary Putnam (1960)
A Computing Procedure for Quantification Theory.
Journal of hte ACM 7(3): 201-215

Martin Davis, George Logemann, & Donald W. Loveland (1962)
A machine program for theorem-proving.
Communications of the ACM 5(7): 394-397



DP Resolution

DPLL

DP Resolution / Variable Elimination [DavisPutnam'60]

Definition (Resolution Rule)

$$\frac{C \vee x \qquad \neg x \vee D}{C \vee D}$$

Resolution on clause sets Γ_x and $\Gamma_{\neg x}$ (denoted by $\Gamma_x \bowtie_x \Gamma_{\neg x}$) generates all non-tautological resolvents of $C \in \Gamma_x$ and $D \in \Gamma_{\neg x}$.

DP Resolution / Variable Elimination [DavisPutnam'60]

Definition (Resolution Rule)

$$\frac{C \vee x \qquad \neg x \vee D}{C \vee D}$$

Resolution on clause sets Γ_x and $\Gamma_{\neg x}$ (denoted by $\Gamma_x \bowtie_x \Gamma_{\neg x}$) generates all non-tautological resolvents of $C \in \Gamma_x$ and $D \in \Gamma_{\neg x}$.

Definition (Variable elimination (VE))

Given a CNF formula Γ, *variable elimination* (or DP resolution) removes a variable x by replacing Γ_x and $\Gamma_{\neg x}$ by $\Gamma_x \bowtie_x \Gamma_{\neg x}$

DP Resolution / Variable Elimination [DavisPutnam'60]

Definition (Resolution Rule)

$$\frac{C \vee x \qquad \neg x \vee D}{C \vee D}$$

Resolution on clause sets Γ_x and $\Gamma_{\neg x}$ (denoted by $\Gamma_x \bowtie_x \Gamma_{\neg x}$) generates all non-tautological resolvents of $C \in \Gamma_x$ and $D \in \Gamma_{\neg x}$.

Definition (Variable elimination (VE))

Given a CNF formula Γ, *variable elimination* (or DP resolution) removes a variable x by replacing Γ_x and $\Gamma_{\neg x}$ by $\Gamma_x \bowtie_x \Gamma_{\neg x}$

Proof procedure [DavisPutnam60]

VE is a complete proof procedure. Applying VE until fixpoint results in either the empty formula (satisfiable) or empty clause (unsatisfiable)

Example VE by clause distribution [DavisPutnam'60]

Definition (Variable elimination (VE))

Given a CNF formula Γ, *variable elimination* (or DP resolution) removes a variable x by replacing Γ_x and $\Gamma_{\neg x}$ by $\Gamma_x \bowtie_x \Gamma_{\neg x}$

Example VE by clause distribution [DavisPutnam'60]

Definition (Variable elimination (VE))

Given a CNF formula Γ, *variable elimination* (or DP resolution) removes a variable x by replacing Γ_x and $\Gamma_{\neg x}$ by $\Gamma_x \bowtie_x \Gamma_{\neg x}$

Example of clause distribution

	Γ_x: $(x \vee c)$	$(x \vee \neg d)$	$(x \vee \neg a \vee \neg b)$
$\Gamma_{\neg x}$: $(\neg x \vee a)$	$(a \vee c)$	$(a \vee \neg d)$	$(a \vee \neg a \vee \neg b)$
$(\neg x \vee b)$	$(b \vee c)$	$(b \vee \neg d)$	$(b \vee \neg a \vee \neg b)$
$(\neg x \vee \neg e \vee f)$	$(c \vee \neg e \vee f)$	$(\neg d \vee \neg e \vee f)$	$(\neg a \vee \neg b \vee \neg e \vee f)$

Example VE by clause distribution [DavisPutnam'60]

Definition (Variable elimination (VE))

Given a CNF formula Γ, *variable elimination* (or DP resolution) removes a variable x by replacing Γ_x and $\Gamma_{\neg x}$ by $\Gamma_x \bowtie_x \Gamma_{\neg x}$

Example of clause distribution

		Γ_x		
		$(x \vee c)$	$(x \vee \neg d)$	$(x \vee \neg a \vee \neg b)$
$\Gamma_{\neg x}$	$(\neg x \vee a)$	$(a \vee c)$	$(a \vee \neg d)$	~~$(a \vee \neg a \vee \neg b)$~~
	$(\neg x \vee b)$	$(b \vee c)$	$(b \vee \neg d)$	~~$(b \vee \neg a \vee \neg b)$~~
	$(\neg x \vee \neg e \vee f)$	$(c \vee \neg e \vee f)$	$(\neg d \vee \neg e \vee f)$	$(\neg a \vee \neg b \vee \neg e \vee f)$

Example VE by clause distribution [DavisPutnam'60]

Definition (Variable elimination (VE))

Given a CNF formula Γ, *variable elimination* (or DP resolution) removes a variable x by replacing Γ_x and $\Gamma_{\neg x}$ by $\Gamma_x \bowtie_x \Gamma_{\neg x}$

Example of clause distribution

	Γ_x: $(x \vee c)$	$(x \vee \neg d)$	$(x \vee \neg a \vee \neg b)$
$\Gamma_{\neg x}$: $(\neg x \vee a)$	$(a \vee c)$	$(a \vee \neg d)$	~~$(a \vee \neg a \vee \neg b)$~~
$(\neg x \vee b)$	$(b \vee c)$	$(b \vee \neg d)$	~~$(b \vee \neg a \vee \neg b)$~~
$(\neg x \vee \neg e \vee f)$	$(c \vee \neg e \vee f)$	$(\neg d \vee \neg e \vee f)$	$(\neg a \vee \neg b \vee \neg e \vee f)$

In the example: $|\Gamma_x \bowtie \Gamma_{\neg x}| > |\Gamma_x| + |\Gamma_{\neg x}|$

Exponential growth of clauses in general

DP Resolution and Pure Literals

Proposition

Given a CNF formula Γ with pure literal p, the effect of applying the pure literal rule on p is the same as the effect of applying DP resolution on p.

True or false?

DP Resolution and Pure Literals

Proposition

Given a CNF formula Γ with pure literal p, the effect of applying the pure literal rule on p is the same as the effect of applying DP resolution on p.

True or false?

Proof.

True. The pure literal rule assign p to true, which has the effect that all clauses containing p are removed. Applying DP resolution on p also removes all clauses containing literal p, because $\Gamma_p \bowtie \Gamma_{\neg p}$ is empty. □

VE by substitution [EenBiere07]

General idea

Detect definitions $x \leftrightarrow \mathrm{DEF}(p_1, \ldots, p_n)$ in the formula and use them to reduce the number of added clauses

VE by substitution [EenBiere07]

General idea

Detect definitions $x \leftrightarrow \mathrm{DEF}(p_1, \ldots, p_n)$ in the formula and use them to reduce the number of added clauses

Possible gates

definition	D_x	$D_{\neg x}$
$\mathrm{AND}(p_1, \ldots, p_n)$	$(x \vee \neg p_1 \vee \cdots \vee \neg p_n)$	$(\neg x \vee p_1), \ldots, (\neg x \vee p_n)$
$\mathrm{OR}(p_1, \ldots, p_n)$	$(x \vee \neg p_1), \ldots, (x \vee \neg p_n)$	$(\neg x \vee p_1 \vee \cdots \vee p_n)$
$\mathrm{ITE}(c, t, f)$	$(x \vee \neg c \vee \neg t), (x \vee c \vee \neg f)$	$(\neg x \vee \neg c \vee t), (\neg x \vee c \vee f)$

VE by substitution [EenBiere07]

General idea

Detect definitions $x \leftrightarrow \mathrm{DEF}(p_1, \ldots, p_n)$ in the formula and use them to reduce the number of added clauses

Possible gates

definition	D_x	$D_{\neg x}$
$\mathrm{AND}(p_1, \ldots, p_n)$	$(x \vee \neg p_1 \vee \cdots \vee \neg p_n)$	$(\neg x \vee p_1), \ldots, (\neg x \vee p_n)$
$\mathrm{OR}(p_1, \ldots, p_n)$	$(x \vee \neg p_1), \ldots, (x \vee \neg p_n)$	$(\neg x \vee p_1 \vee \cdots \vee p_n)$
$\mathrm{ITE}(c, t, f)$	$(x \vee \neg c \vee \neg t), (x \vee c \vee \neg f)$	$(\neg x \vee \neg c \vee t), (\neg x \vee c \vee f)$

Variable elimination by substitution [EenBiere07]

Let $R_x = \Gamma_x \setminus D_x$; $R_{\neg x} = \Gamma_{\neg x} \setminus D_{\neg x}$.

Replace $\Gamma_x \wedge \Gamma_{\neg x}$ by $D_x \bowtie_x R_{\neg x} \wedge D_{\neg x} \bowtie_x R_x$.

Always less than $\Gamma_x \bowtie_x \Gamma_{\neg x}$! (if x is a definition)

VE by substitution [EenBiere'07]

Example of gate extraction: $x = \mathrm{AND}(a, b)$

$$\Gamma_x = (x \vee c) \wedge (x \vee \neg d) \wedge (x \vee \neg a \vee \neg b)$$
$$\Gamma_{\neg x} = (\neg x \vee a) \wedge (\neg x \vee b) \wedge (\neg x \vee \neg e \vee f)$$

VE by substitution [EenBiere'07]

Example of gate extraction: $x = \mathrm{AND}(a, b)$

$$\Gamma_x = (x \vee c) \wedge (x \vee \neg d) \wedge (x \vee \neg a \vee \neg b)$$
$$\Gamma_{\neg x} = (\neg x \vee a) \wedge (\neg x \vee b) \wedge (\neg x \vee \neg e \vee f)$$

Example of substitution

		R_x		D_x
		$(x \vee c)$	$(x \vee \neg d)$	$(x \vee \neg a \vee \neg b)$
$D_{\neg x}$	$(\neg x \vee a)$	$(a \vee c)$	$(a \vee \neg d)$	
	$(\neg x \vee b)$	$(b \vee c)$	$(b \vee \neg d)$	
$R_{\neg x}$	$(\neg x \vee \neg e \vee f)$			$(\neg a \vee \neg b \vee \neg e \vee f)$

VE by substitution [EenBiere'07]

Example of gate extraction: $x = \mathrm{AND}(a, b)$

$$\Gamma_x = (x \vee c) \wedge (x \vee \neg d) \wedge (x \vee \neg a \vee \neg b)$$
$$\Gamma_{\neg x} = (\neg x \vee a) \wedge (\neg x \vee b) \wedge (\neg x \vee \neg e \vee f)$$

Example of substitution

		R_x		D_x
		$(x \vee c)$	$(x \vee \neg d)$	$(x \vee \neg a \vee \neg b)$
$D_{\neg x}$	$(\neg x \vee a)$	$(a \vee c)$	$(a \vee \neg d)$	
	$(\neg x \vee b)$	$(b \vee c)$	$(b \vee \neg d)$	
$R_{\neg x}$	$(\neg x \vee \neg e \vee f)$			$(\neg a \vee \neg b \vee \neg e \vee f)$

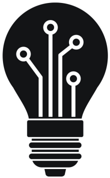
using substitution: $|\Gamma_x \bowtie \Gamma_{\neg x}| < |\Gamma_x| + |\Gamma_{\neg x}|$

DP Resolution

DPLL

SAT Solver Paradigms Overview

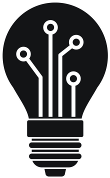
DPLL: Aims at finding a small search-tree by selecting effective splitting variables (e.g. via looking ahead).



Strength: Effective on small, hard formulas.

Weakness: Expensive.

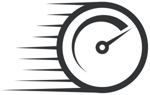
SAT Solver Paradigms Overview



DPLL: Aims at finding a small search-tree by selecting effective splitting variables (e.g. via looking ahead).

Strength: Effective on small, hard formulas.

Weakness: Expensive.



Local search: Given a full assignment for a formula Γ, flip the truth values of variables until satisfying Γ.

Strength: Can quickly find solutions for hard formulas.

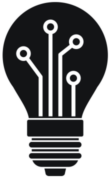
Weakness: Cannot prove unsatisfiability.

SAT Solver Paradigms Overview

DPLL: Aims at finding a small search-tree by selecting effective splitting variables (e.g. via looking ahead).

Strength: Effective on small, hard formulas.

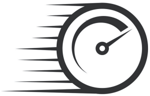
Weakness: Expensive.



Local search: Given a full assignment for a formula Γ, flip the truth values of variables until satisfying Γ.

Strength: Can quickly find solutions for hard formulas.

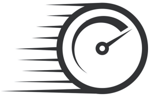
Weakness: Cannot prove unsatisfiability.



Conflict-driven clause learning (CDCL): Makes fast decisions and converts conflicts into learned clauses.

Strength: Effective on large, "easy" formulas.

Weakness: Hard to parallelize.



DPLL: Introduction

Davis Putnam Logemann Loveland [DP60,DLL62]

Recursive procedure that in each recursive call:

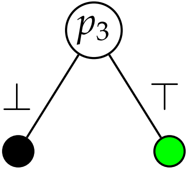
- Simplifies the formula (using unit propagation)
- Splits the formula into two subformulas
  - Variable selection heuristics (which variable to split on)
  - Direction heuristics (which subformula to explore first)

DPLL: Example

$$\Gamma_{\text{DPLL}} := (p_1 \vee p_2 \vee \neg p_3) \wedge (\neg p_1 \vee p_2 \vee p_3) \wedge (\neg p_1 \vee \neg p_2 \vee p_3) \wedge (p_1 \vee p_3) \wedge (\neg p_1 \vee \neg p_3)$$

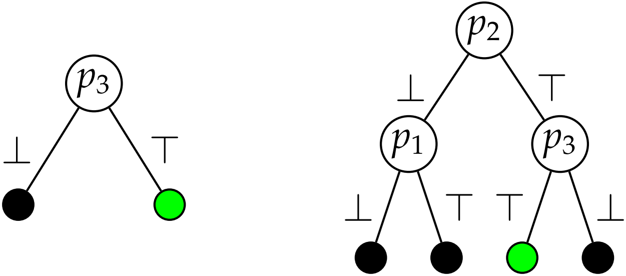
DPLL: Example

$$\Gamma_{\text{DPLL}} := (p_1 \vee p_2 \vee \neg p_3) \wedge (\neg p_1 \vee p_2 \vee p_3) \wedge (\neg p_1 \vee \neg p_2 \vee p_3) \wedge (p_1 \vee p_3) \wedge (\neg p_1 \vee \neg p_3)$$



DPLL: Example

$$\Gamma_{\text{DPLL}} := (p_1 \vee p_2 \vee \neg p_3) \wedge (\neg p_1 \vee p_2 \vee p_3) \wedge (\neg p_1 \vee \neg p_2 \vee p_3) \wedge (p_1 \vee p_3) \wedge (\neg p_1 \vee \neg p_3)$$



DPLL: Slightly Harder Example

Construct a DPLL tree for:

$$(p \vee q \vee \neg r) \wedge (\neg p \vee \neg q \vee r) \wedge$$
$$(q \vee r \vee \neg s) \wedge (\neg q \vee \neg r \vee s) \wedge$$
$$(p \vee r \vee s) \wedge (\neg p \vee \neg r \vee \neg s) \wedge$$
$$(\neg p \vee q \vee s)$$

What is a good heuristic?

DPLL: Slightly Harder Example

Construct a DPLL tree for:

$$(p \lor q \lor \neg r) \land (\neg p \lor \neg q \lor r) \land \\ (q \lor r \lor \neg s) \land (\neg q \lor \neg r \lor s) \land \\ (p \lor r \lor s) \land (\neg p \lor \neg r \lor \neg s) \land \\ (\neg p \lor q \lor s)$$

What is a good heuristic?

A cheap and reasonably effective heuristic is MOMS:
Maximum Occurrence in clauses of Minimum Size

DPLL: Pseudocode

DPLL (τ, Γ)

$\tau' :=$ Simplify (τ, Γ)

if $[\![\Gamma]\!]_{\tau'} = \top$ **then return** `satisfiable`

if $[\![\Gamma]\!]_{\tau'} = \bot$ **then return** `unsatisfiable`

$\ell_{\text{decision}} :=$ Decide (τ', Γ)

if (DPLL($\tau' \cup \ell_{\text{decision}} := \top, \Gamma)$ = `satisfiable`) **then**

    return `satisfiable`

return DPLL $(\tau' \cup \ell_{\text{decision}} := \bot, \Gamma)$

DPLL: Demo in Lean

LAMR/Examples/using_sat_solvers/dpll.lean

```
partial def dpllSatAux (τ : PropAssignment) (Γ : CnfForm) :
    Option (PropAssignment × CnfForm) :=
  if Γ.hasEmpty then none
  else match pickSplit? Γ with
  -- No variables left to split on, we found a solution.
  | none => some (τ, Γ)
  -- Split on `x`.
  -- `<|>` is the "or else" operator, which tries one action and if that fails
  -- tries the other.
  | some x => goWithNew x τ Γ <|> goWithNew (-x) τ Γ

where
  /-- Assigns `x` to true and continues out DPLL. -/
  goWithNew (x : Lit) (τ : PropAssignment) (Γ : CnfForm) :
      Option (PropAssignment × CnfForm) :=
    let (τ', Γ') := propagateWithNew x τ Γ
    dpllSatAux τ' Γ'

/-- Solve `Γ` using DPLL. Return a satisfying assignment if found, otherwise `none`. -/
def dpllSat (Γ : CnfForm) : Option PropAssignment :=
  let ⟨τ, Γ⟩ := propagateUnits [] Γ
  (dpllSatAux τ Γ).map fun ⟨τ, _⟩ => τ
```

DPLL: Look-aheads

DPLL with selection of (effective) decision variables by look-aheads on variables

DPLL: Look-aheads

DPLL with selection of (effective) decision variables by look-aheads on variables

Look-ahead:

- Assign a variable to a truth value

DPLL: Look-aheads

DPLL with selection of (effective) decision variables by look-aheads on variables

Look-ahead:

- Assign a variable to a truth value
- Simplify the formula

DPLL: Look-aheads

DPLL with selection of (effective) decision variables by look-aheads on variables

Look-ahead:

- Assign a variable to a truth value
- Simplify the formula
- Measure the reduction

DPLL: Look-aheads

DPLL with selection of (effective) decision variables by look-aheads on variables

Look-ahead:

- Assign a variable to a truth value
- Simplify the formula
- Measure the reduction
- Learn if possible

DPLL: Look-aheads

DPLL with selection of (effective) decision variables by look-aheads on variables

Look-ahead:

- Assign a variable to a truth value
- Simplify the formula
- Measure the reduction
- Learn if possible
- Backtrack

DPLL: Look-ahead Reduction Heuristics

- Number of satisfied clauses

DPLL: Look-ahead Reduction Heuristics

- Number of satisfied clauses
- Number of implied variables

DPLL: Look-ahead Reduction Heuristics

- Number of satisfied clauses
- Number of implied variables
- New (reduced, not satisfied) clauses
  - Smaller clauses more important
  - Weights based on occurrences

DPLL: Learning Necessary Assignments

$$\Gamma_{\text{LEARN}} := (\neg p_1 \vee \neg p_3 \vee p_4) \wedge (\neg p_1 \vee \neg p_2 \vee p_3) \wedge (\neg p_1 \vee p_2) \wedge (p_1 \vee p_3 \vee p_6) \wedge (\neg p_1 \vee p_4 \vee \neg p_5) \wedge (p_1 \vee \neg p_6) \wedge (p_4 \vee p_5 \vee p_6) \wedge (p_5 \vee \neg p_6)$$

DPLL: Learning Necessary Assignments

$$\Gamma_{\text{LEARN}} := (\neg p_1 \vee \neg p_3 \vee p_4) \wedge (\neg p_1 \vee \neg p_2 \vee p_3) \wedge (\neg p_1 \vee p_2) \wedge (p_1 \vee p_3 \vee p_6) \wedge (\neg p_1 \vee p_4 \vee \neg p_5) \wedge (p_1 \vee \neg p_6) \wedge (p_4 \vee p_5 \vee p_6) \wedge (p_5 \vee \neg p_6)$$

$$\tau = \{p_1 = \top\}$$

DPLL: Learning Necessary Assignments

$$\Gamma_{\text{LEARN}} := (\neg p_1 \vee \neg p_3 \vee p_4) \wedge (\neg p_1 \vee \neg p_2 \vee p_3) \wedge (\neg p_1 \vee p_2) \wedge (p_1 \vee p_3 \vee p_6) \wedge (\neg p_1 \vee p_4 \vee \neg p_5) \wedge (p_1 \vee \neg p_6) \wedge (p_4 \vee p_5 \vee p_6) \wedge (p_5 \vee \neg p_6)$$

$$\tau = \{p_1 = \top, p_2 = \top\}$$

DPLL: Learning Necessary Assignments

$$\Gamma_{\text{LEARN}} := (\neg p_1 \vee \neg p_3 \vee p_4) \wedge (\neg p_1 \vee \neg p_2 \vee p_3) \wedge (\neg p_1 \vee p_2) \wedge (p_1 \vee p_3 \vee p_6) \wedge (\neg p_1 \vee p_4 \vee \neg p_5) \wedge (p_1 \vee \neg p_6) \wedge (p_4 \vee p_5 \vee p_6) \wedge (p_5 \vee \neg p_6)$$

$$\tau = \{p_1 = \top, p_2 = \top, p_3 = \top\}$$

DPLL: Learning Necessary Assignments

$$\Gamma_{\text{LEARN}} := (\neg p_1 \vee \neg p_3 \vee p_4) \wedge (\neg p_1 \vee \neg p_2 \vee p_3) \wedge (\neg p_1 \vee p_2) \wedge (p_1 \vee p_3 \vee p_6) \wedge (\neg p_1 \vee p_4 \vee \neg p_5) \wedge (p_1 \vee \neg p_6) \wedge (p_4 \vee p_5 \vee p_6) \wedge (p_5 \vee \neg p_6)$$

$$\tau = \{p_1 = \top, p_2 = \top, p_3 = \top, p_4 = \top\}$$

DPLL: Learning Necessary Assignments

$$\Gamma_{\text{LEARN}} := (\neg p_1 \vee \neg p_3 \vee p_4) \wedge (\neg p_1 \vee \neg p_2 \vee p_3) \wedge (\neg p_1 \vee p_2) \wedge (p_1 \vee p_3 \vee p_6) \wedge (\neg p_1 \vee p_4 \vee \neg p_5) \wedge (p_1 \vee \neg p_6) \wedge (p_4 \vee p_5 \vee p_6) \wedge (p_5 \vee \neg p_6)$$

$$\tau = \{p_1 = \top, p_2 = \top, p_3 = \top, p_4 = \top\}$$

$$\Gamma_{\text{LEARN}} := (\neg p_1 \vee \neg p_3 \vee p_4) \wedge (\neg p_1 \vee \neg p_2 \vee p_3) \wedge (\neg p_1 \vee p_2) \wedge (p_1 \vee p_3 \vee p_6) \wedge (\neg p_1 \vee p_4 \vee \neg p_5) \wedge (p_1 \vee \neg p_6) \wedge (p_4 \vee p_5 \vee p_6) \wedge (p_5 \vee \neg p_6)$$

DPLL: Learning Necessary Assignments

$$\Gamma_{\text{LEARN}} := (\neg p_1 \vee \neg p_3 \vee p_4) \wedge (\neg p_1 \vee \neg p_2 \vee p_3) \wedge (\neg p_1 \vee p_2) \wedge (p_1 \vee p_3 \vee p_6) \wedge (\neg p_1 \vee p_4 \vee \neg p_5) \wedge (p_1 \vee \neg p_6) \wedge (p_4 \vee p_5 \vee p_6) \wedge (p_5 \vee \neg p_6)$$

$$\tau = \{p_1 = \top, p_2 = \top, p_3 = \top, p_4 = \top\}$$

$$\Gamma_{\text{LEARN}} := (\neg p_1 \vee \neg p_3 \vee p_4) \wedge (\neg p_1 \vee \neg p_2 \vee p_3) \wedge (\neg p_1 \vee p_2) \wedge (p_1 \vee p_3 \vee p_6) \wedge (\neg p_1 \vee p_4 \vee \neg p_5) \wedge (p_1 \vee \neg p_6) \wedge (p_4 \vee p_5 \vee p_6) \wedge (p_5 \vee \neg p_6)$$

$$\tau = \{p_1 = \bot\}$$

DPLL: Learning Necessary Assignments

$$\Gamma_{\text{LEARN}} := (\neg p_1 \vee \neg p_3 \vee p_4) \wedge (\neg p_1 \vee \neg p_2 \vee p_3) \wedge (\neg p_1 \vee p_2) \wedge (p_1 \vee p_3 \vee p_6) \wedge (\neg p_1 \vee p_4 \vee \neg p_5) \wedge (p_1 \vee \neg p_6) \wedge (p_4 \vee p_5 \vee p_6) \wedge (p_5 \vee \neg p_6)$$

$$\tau = \{p_1 = \top, p_2 = \top, p_3 = \top, p_4 = \top\}$$

$$\Gamma_{\text{LEARN}} := (\neg p_1 \vee \neg p_3 \vee p_4) \wedge (\neg p_1 \vee \neg p_2 \vee p_3) \wedge (\neg p_1 \vee p_2) \wedge (p_1 \vee p_3 \vee p_6) \wedge (\neg p_1 \vee p_4 \vee \neg p_5) \wedge (p_1 \vee \neg p_6) \wedge (p_4 \vee p_5 \vee p_6) \wedge (p_5 \vee \neg p_6)$$

$$\tau = \{p_1 = \bot, p_6 = \bot\}$$

DPLL: Learning Necessary Assignments

$$\Gamma_{\text{LEARN}} := (\neg p_1 \vee \neg p_3 \vee p_4) \wedge (\neg p_1 \vee \neg p_2 \vee p_3) \wedge (\neg p_1 \vee p_2) \wedge (p_1 \vee p_3 \vee p_6) \wedge (\neg p_1 \vee p_4 \vee \neg p_5) \wedge (p_1 \vee \neg p_6) \wedge (p_4 \vee p_5 \vee p_6) \wedge (p_5 \vee \neg p_6)$$

$$\tau = \{p_1 = \top, p_2 = \top, p_3 = \top, p_4 = \top\}$$

$$\Gamma_{\text{LEARN}} := (\neg p_1 \vee \neg p_3 \vee p_4) \wedge (\neg p_1 \vee \neg p_2 \vee p_3) \wedge (\neg p_1 \vee p_2) \wedge (p_1 \vee p_3 \vee p_6) \wedge (\neg p_1 \vee p_4 \vee \neg p_5) \wedge (p_1 \vee \neg p_6) \wedge (p_4 \vee p_5 \vee p_6) \wedge (p_5 \vee \neg p_6)$$

$$\tau = \{p_1 = \bot, p_6 = \bot, p_3 = \top\}$$

DPLL: Look-ahead Autarky Detection

$$\Gamma_{\text{LEARN}} := (\neg p_1 \vee \neg p_3 \vee p_4) \wedge (\neg p_1 \vee \neg p_2 \vee p_3) \wedge (\neg p_1 \vee p_2) \wedge (p_1 \vee p_3 \vee p_6) \wedge (\neg p_1 \vee p_4 \vee \neg p_5) \wedge (p_1 \vee \neg p_6) \wedge (p_4 \vee p_5 \vee p_6) \wedge (p_5 \vee \neg p_6)$$

DPLL: Look-ahead Autarky Detection

$$\Gamma_{\text{LEARN}} := (\neg p_1 \vee \neg p_3 \vee p_4) \wedge (\neg p_1 \vee \neg p_2 \vee p_3) \wedge (\neg p_1 \vee p_2) \wedge (p_1 \vee p_3 \vee p_6) \wedge (\neg p_1 \vee p_4 \vee \neg p_5) \wedge (p_1 \vee \neg p_6) \wedge (p_4 \vee p_5 \vee p_6) \wedge (p_5 \vee \neg p_6)$$

$\tau = \{p_1 = \top\}$

DPLL: Look-ahead Autarky Detection

$$\Gamma_{\text{LEARN}} := (\neg p_1 \vee \neg p_3 \vee p_4) \wedge (\neg p_1 \vee \neg p_2 \vee p_3) \wedge (\neg p_1 \vee p_2) \wedge (p_1 \vee p_3 \vee p_6) \wedge (\neg p_1 \vee p_4 \vee \neg p_5) \wedge (p_1 \vee \neg p_6) \wedge (p_4 \vee p_5 \vee p_6) \wedge (p_5 \vee \neg p_6)$$

$$\tau = \{p_1 = \top, p_2 = \top\}$$

DPLL: Look-ahead Autarky Detection

$$\Gamma_{\text{LEARN}} := (\neg p_1 \vee \neg p_3 \vee p_4) \wedge (\neg p_1 \vee \neg p_2 \vee p_3) \wedge (\neg p_1 \vee p_2) \wedge (p_1 \vee p_3 \vee p_6) \wedge (\neg p_1 \vee p_4 \vee \neg p_5) \wedge (p_1 \vee \neg p_6) \wedge (p_4 \vee p_5 \vee p_6) \wedge (p_5 \vee \neg p_6)$$

$$\tau = \{p_1 = \top, p_2 = \top, p_3 = \top\}$$

DPLL: Look-ahead Autarky Detection

$$\Gamma_{\text{LEARN}} := (\neg p_1 \vee \neg p_3 \vee p_4) \wedge (\neg p_1 \vee \neg p_2 \vee p_3) \wedge (\neg p_1 \vee p_2) \wedge (p_1 \vee p_3 \vee p_6) \wedge (\neg p_1 \vee p_4 \vee \neg p_5) \wedge (p_1 \vee \neg p_6) \wedge (p_4 \vee p_5 \vee p_6) \wedge (p_5 \vee \neg p_6)$$

$$\tau = \{p_1 = \top, p_2 = \top, p_3 = \top, p_4 = \top\}$$

DPLL: Look-ahead Autarky Detection

$$\Gamma_{\text{LEARN}} := (\neg p_1 \vee \neg p_3 \vee p_4) \wedge (\neg p_1 \vee \neg p_2 \vee p_3) \wedge (\neg p_1 \vee p_2) \wedge (p_1 \vee p_3 \vee p_6) \wedge (\neg p_1 \vee p_4 \vee \neg p_5) \wedge (p_1 \vee \neg p_6) \wedge (p_4 \vee p_5 \vee p_6) \wedge (p_5 \vee \neg p_6)$$

$$\tau = \{p_1 = \top, p_2 = \top, p_3 = \top, p_4 = \top\}$$

Γ_{LEARN} satisfiability equivalent to $(p_5 \vee \neg p_6)$

DPLL: Look-ahead Autarky Detection

$$\Gamma_{\text{LEARN}} := (\neg p_1 \vee \neg p_3 \vee p_4) \wedge (\neg p_1 \vee \neg p_2 \vee p_3) \wedge (\neg p_1 \vee p_2) \wedge (p_1 \vee p_3 \vee p_6) \wedge (\neg p_1 \vee p_4 \vee \neg p_5) \wedge (p_1 \vee \neg p_6) \wedge (p_4 \vee p_5 \vee p_6) \wedge (p_5 \vee \neg p_6)$$

$$\tau = \{p_1 = \top, p_2 = \top, p_3 = \top, p_4 = \top\}$$

Γ_{LEARN} satisfiability equivalent to $(p_5 \vee \neg p_6)$

Could reduce computational cost on UNSAT

DPLL: Look-ahead 1-Autarky Learning

$$\Gamma_{\text{LEARN}} := (\neg p_1 \vee \neg p_3 \vee p_4) \wedge (\neg p_1 \vee \neg p_2 \vee p_3) \wedge (\neg p_1 \vee p_2) \wedge (p_1 \vee p_3 \vee p_6) \wedge (\neg p_1 \vee p_4 \vee \neg p_5) \wedge (p_1 \vee \neg p_6) \wedge (p_4 \vee p_5 \vee p_6) \wedge (p_5 \vee \neg p_6)$$

DPLL: Look-ahead 1-Autarky Learning

$$\Gamma_{\text{LEARN}} := (\neg p_1 \vee \neg p_3 \vee p_4) \wedge (\neg p_1 \vee \neg p_2 \vee p_3) \wedge (\neg p_1 \vee p_2) \wedge (p_1 \vee p_3 \vee p_6) \wedge (\neg p_1 \vee p_4 \vee \neg p_5) \wedge (p_1 \vee \neg p_6) \wedge (p_4 \vee p_5 \vee p_6) \wedge (p_5 \vee \neg p_6)$$

$$\tau = \{p_2 = \bot\}$$

DPLL: Look-ahead 1-Autarky Learning

$$\Gamma_{\text{LEARN}} := (\neg p_1 \vee \neg p_3 \vee p_4) \wedge (\neg p_1 \vee \neg p_2 \vee p_3) \wedge (\neg p_1 \vee p_2) \wedge (p_1 \vee p_3 \vee p_6) \wedge (\neg p_1 \vee p_4 \vee \neg p_5) \wedge (p_1 \vee \neg p_6) \wedge (p_4 \vee p_5 \vee p_6) \wedge (p_5 \vee \neg p_6)$$

$$\tau = \{p_2 = \bot, p_1 = \bot\}$$

DPLL: Look-ahead 1-Autarky Learning

$$\Gamma_{\text{LEARN}} := (\neg p_1 \vee \neg p_3 \vee p_4) \wedge (\neg p_1 \vee \neg p_2 \vee p_3) \wedge (\neg p_1 \vee p_2) \wedge (p_1 \vee p_3 \vee p_6) \wedge (\neg p_1 \vee p_4 \vee \neg p_5) \wedge (p_1 \vee \neg p_6) \wedge (p_4 \vee p_5 \vee p_6) \wedge (p_5 \vee \neg p_6)$$

$$\tau = \{p_2 = \bot, p_1 = \bot, p_6 = \bot\}$$

DPLL: Look-ahead 1-Autarky Learning

$$\Gamma_{\text{LEARN}} := (\neg p_1 \vee \neg p_3 \vee p_4) \wedge (\neg p_1 \vee \neg p_2 \vee p_3) \wedge (\neg p_1 \vee p_2) \wedge (p_1 \vee p_3 \vee p_6) \wedge (\neg p_1 \vee p_4 \vee \neg p_5) \wedge (p_1 \vee \neg p_6) \wedge (p_4 \vee p_5 \vee p_6) \wedge (p_5 \vee \neg p_6)$$

$$\tau = \{p_2 = \bot, p_1 = \bot, p_6 = \bot, p_3 = \top\}$$

DPLL: Look-ahead 1-Autarky Learning

$$\Gamma_{\text{LEARN}} := (\neg p_1 \vee \neg p_3 \vee p_4) \wedge (\neg p_1 \vee \neg p_2 \vee p_3) \wedge (\neg p_1 \vee p_2) \wedge (p_1 \vee p_3 \vee p_6) \wedge (\neg p_1 \vee p_4 \vee \neg p_5) \wedge (p_1 \vee \neg p_6) \wedge (p_4 \vee p_5 \vee p_6) \wedge (p_5 \vee \neg p_6)$$

$$\tau = \{p_2 = \bot, p_1 = \bot, p_6 = \bot, p_3 = \top\}$$

(local) 1-autarky resolvents to add to Γ_{LEARN}:
$(\neg p_2 \vee \neg p_4)$ and $(\neg p_2 \vee \neg p_5)$

DPLL: Complexity

Can $n+1$ pigeons be in n holes (at-most-one pigeon per hole)?

$$PHP_n := \bigwedge_{1 \leq p \leq n+1} (x_{1,p} \vee \cdots \vee x_{n,p}) \wedge \bigwedge_{1 \leq h \leq n,\, 1 \leq p < q \leq n+1} (\overline{x}_{h,p} \vee \overline{x}_{h,q})$$

Resolution proofs of PHP_n are exponential [Haken 1985]

Cook constructed polynomial-sized ER proofs of PHP_n [1976]

- Requires auxiliary variables

DPLL: Complexity

Can $n+1$ pigeons be in n holes (at-most-one pigeon per hole)?

$$PHP_n := \bigwedge_{1 \le p \le n+1} (x_{1,p} \vee \cdots \vee x_{n,p}) \wedge \bigwedge_{1 \le h \le n,\, 1 \le p < q \le n+1} (\overline{x}_{h,p} \vee \overline{x}_{h,q})$$

Resolution proofs of PHP_n are exponential [Haken 1985]

Cook constructed polynomial-sized ER proofs of PHP_n [1976]

- Requires auxiliary variables

Polynomial-sized conditional autarky proofs of PHP_n

- Without auxiliary variables