Decision Procedures and Verification

NAIL094

Petr Kučera

Charles University

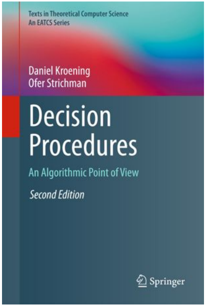
2019/20 (1st lecture)

Introduction

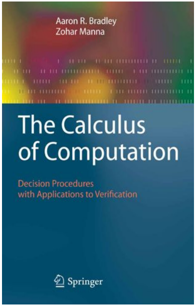
Overview

- Introduction, motivation, basic notions.
- Normal forms: CNF, DNF, NNF
- Satisfiability of boolean formulas
  - Modern SAT solvers
  - Local search
  - binary decision diagrams (BDD)
  - quantified boolean formulas (QBF)
- Decision procedures for theories
  - equality and uninterpreted functions
  - linear arithmetic
  - bit vectors
  - arrays, memory, pointers
- Combination of theories
- …

Literature

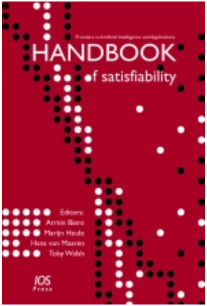


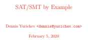
Kroening, D., & Strichman, O. (2016). Decision procedures. Springer.



Bradley, A. R., & Manna, Z. (2007). The calculus of computation. Springer.

Literature




SAT/SMT by Example

Dennis Yurichev <dennis@yurichev.com>

February 5, 2020

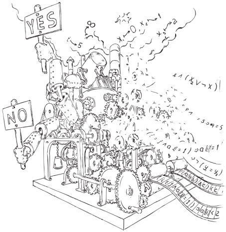
Biere, A., Heule, M., & van Maaren, H. (Eds.). (2009). Handbook of satisfiability. IOS press.

Dennis Yurichev. (2020). SAT/SMT by Example

Decision Procedure

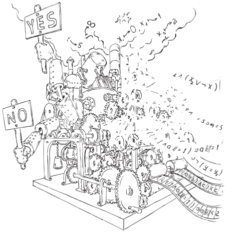
Intuition

A decision procedure is an algorithm that, given a logical formula, decides if it is satisfiable.



Picture by Ilya Yodovsky Jr. (taken from Decision Procedures by D. Kroening and O. Strichman)

Decision Procedure



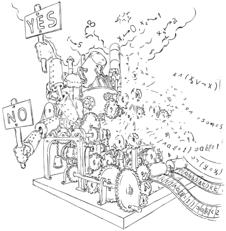
Intuition

A decision procedure is an algorithm that, given a logical formula, decides if it is satisfiable.

Satisfiable $\longrightarrow$ satisfying assignment (model)

Picture by Ilya Yodovsky Jr. (taken from Decision Procedures by D. Kroening and O. Strichman)

Decision Procedure



Intuition

A decision procedure is an algorithm that, given a logical formula, decides if it is satisfiable.

Satisfiable $\longrightarrow$ satisfying assignment (model)
Unsatisfiable $\longrightarrow$ proof of unsatisfiability

Picture by Ilya Yodovsky Jr. (taken from Decision Procedures by D. Kroening and O. Strichman)

Applications

Everywhere where logic is the primary modelling language

- Model checking
- Hardware verification
  - Verifying designs of electronic circuits
- Software verification
  - Verifying that an assertion in code cannot be violated
- Compiler optimizations
  - Correctness of transformations
- Software package dependencies
  - Dependency hell
- Planning and scheduling
- Chemical reaction networks
- …

Propositional SAT Solving

Language of Propositional Logic

- Countable set of propositional variables $\{x_0, x_2, \dots\}$
- Logical connectives $\{\neg, \wedge, \vee, \rightarrow, \leftrightarrow\}$

Definition (Propositional formula)

- A variable is a formula
- If φ and ψ are formulas, then $\neg\varphi$, $\varphi \wedge \psi$, $\varphi \vee \psi$, $\varphi \rightarrow \psi$, and $\varphi \leftrightarrow \psi$ are formulas
- Nothing else is a formula

Assignments and Satisfaction

- An assignment $\mathbf{a}$ maps propositional variables to true or false (1 or 0, $\top$ or $\bot$)
- An assignment $\mathbf{a}$ satisfies formula φ if $\varphi(\mathbf{a})$ evaluates to true
- We also say it is a model of φ
- We also write $\alpha \models \varphi$
- A formula is satisfiable if there exists an assignment that satisfies it
- A formula is a contradiction or unsatisfiable otherwise
- A formula is valid (tautology) if it is satisfied by every assignment

Formula φ is a tautology if and only if $\neg\varphi$ is not satisfiable

Elements of Normal Forms

Literal a propositional variable x or its negation $\neg x$

Term a conjunction of literals, e.g. $x \wedge \neg y \wedge \neg z$

Clause a disjunction of literals, e.g. $x \vee \neg y \vee \neg z$

NNF a formula is in negation normal form if it uses only $\wedge$, $\vee$, $\neg$. Negation only in front of variables (in literals)

CNF a formula is in conjunctive normal form if it is a conjunction of clauses

DNF a formula is in disjunctive normal form if it is a disjunction of clauses

- The empty term and empty CNF are valid — $\top$
- The empty clause and empty DNF are contradictions — $\perp$

SAT and Related Problems

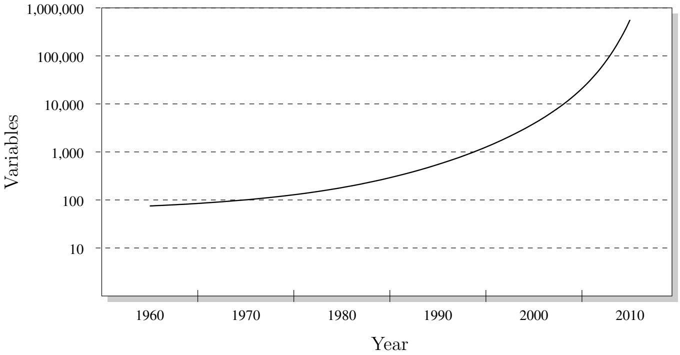
Given a CNF formula φ

- **SAT** Is φ satisfiable?
- **UNSAT** Is φ unsatisfiable?
- **MaxSAT** Find an assignment maximizing the number of satisfied clauses of φ
- **#SAT** Count the models of φ
- **Model Enumeration** Enumerate the models of φ

Why Study Propositional SAT?

- Interesting from both theoretical and practical perspective
- The first problem to be proven NP-complete
  - Cook, 1971; Levin, 1973
- Generic problems — many problems encoded into SAT
  - Hardware and software verification
  - Planning and scheduling
  - Product configuration
  - …and many others

Progress in SAT Solving



The size of industrial CNF formulas that are regularly solved by SAT solvers in a few hours, according to year.

Image source: Decision Procedures. Kroening D., Strichman O.

Progress in SAT Solving

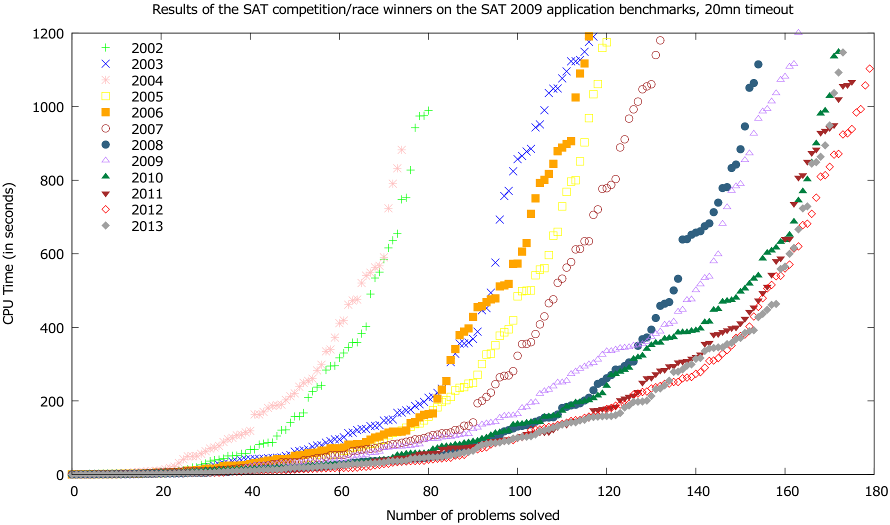
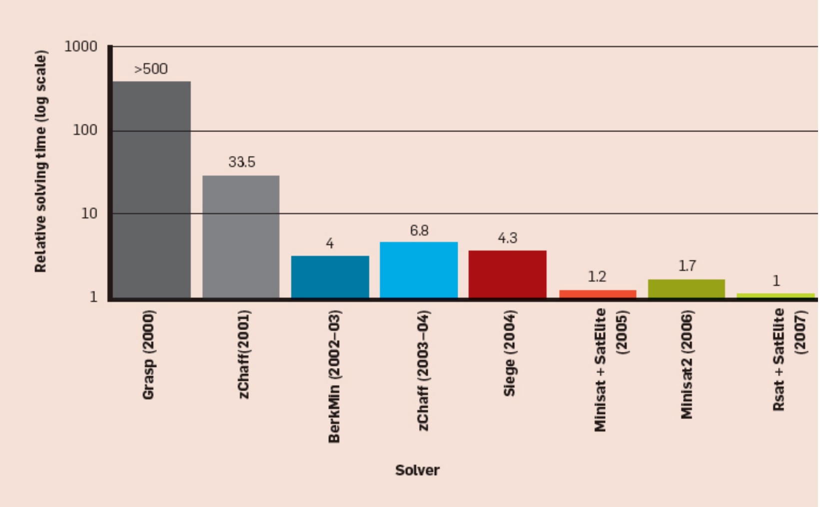


Image source: Decision Procedures. Kroening D., Strichman O.

Progress in SAT Solving



Sharad Malik, Lintao Zhang Communications of the ACM, August 2009, Vol. 52 No. 8, Pages 76–82

Converting to an Equivalent CNF

Lemma

Every formula can be modified to an equivalent CNF.

Construction

1. Convert to NNF
   a. Rewrite connectives using only $\wedge$, $\vee$, and $\neg$
   b. Use De Morgan's laws to propagate negations to variables
   c. Remove double negation ($\neg\neg x = x$)
2. Use distributivity to propagate disjunction over conjunction

The result can be exponentially larger!

$$\bigvee_{i=1}^{n} (x_i \wedge y_i)$$

Tseitin's Encoding

Lemma (Tseitin)

Every formula can be converted to an equisatisfiable formula in CNF which is larger only by a constant factor.

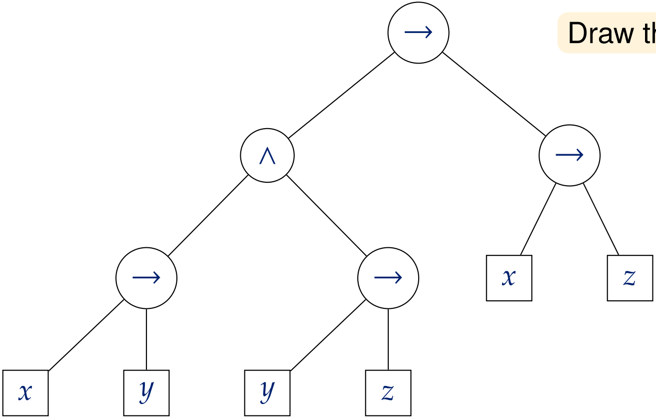
Idea:

- Draw a derivation tree of the formula
- Assign a fresh variable to each connective
- Add clauses to define the function of each connective
- Add the root variable as a unit clause

Can be used to convert a logical circuit into a CNF as well.

Tseitin's Encoding (example)

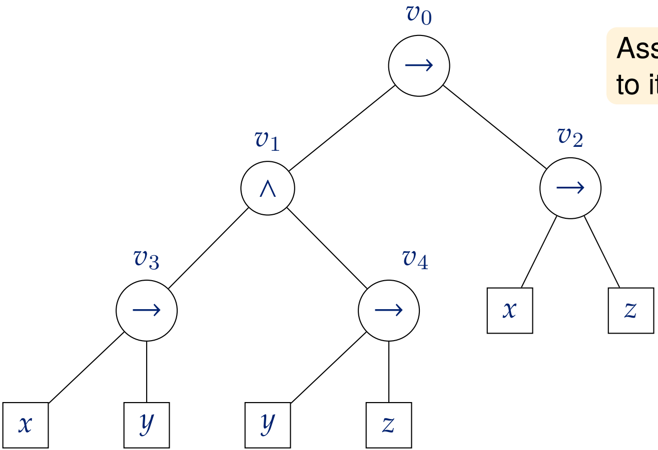
$$(x \to y) \wedge (y \to z) \to (x \to z)$$



Draw the derivation tree

Tseitin's Encoding (example)

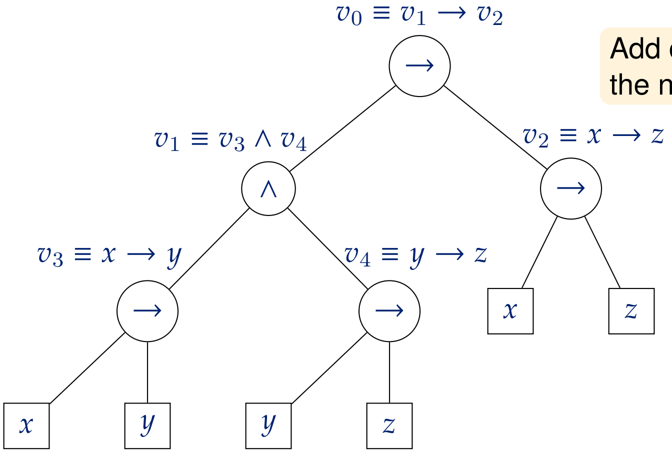
$$(x \to y) \land (y \to z) \to (x \to z)$$



Assign variables to its nodes

Tseitin's Encoding (example)

$$(x \rightarrow y) \wedge (y \rightarrow z) \rightarrow (x \rightarrow z)$$



Add definitions of the new variables

Tseitin's Encoding (example)

$$(x \to y) \wedge (y \to z) \to (x \to z)$$

- rewrite as a conjunction of definitions of new variables

$$\begin{aligned}
&v_0 \\
&v_0 \equiv v_1 \to v_2 \\
&v_1 \equiv v_3 \wedge v_4 \\
&v_2 \equiv x \to z \\
&v_3 \equiv x \to y \\
&v_3 \equiv y \to z
\end{aligned}$$

Tseitin's Encoding (example)

$$(x \rightarrow y) \wedge (y \rightarrow z) \rightarrow (x \rightarrow z)$$

- rewrite as a conjunction of definitions of new variables
- rewrite definitions only using $\neg$, $\wedge$, and $\vee$

$$\begin{aligned}
&v_0 \\
&v_0 \equiv v_1 \longrightarrow v_2 \equiv \neg v_1 \vee v_2 \\
&v_1 \equiv v_3 \wedge v_4 \\
&v_2 \equiv x \longrightarrow z \equiv \neg x \vee z \\
&v_3 \equiv x \longrightarrow y \equiv \neg x \vee y \\
&v_3 \equiv y \longrightarrow z \equiv \neg y \vee z
\end{aligned}$$

Tseitin's Encoding (example)

$$(x \to y) \land (y \to z) \to (x \to z)$$

- rewrite as a conjunction of definitions of new variables
- rewrite definitions only using $\neg$, $\land$, and $\lor$

$$\begin{aligned}
&v_0 \\
&v_0 \equiv \neg v_1 \lor v_2 \\
&v_1 \equiv v_3 \land v_4 \\
&v_2 \equiv \neg x \lor z \\
&v_3 \equiv \neg x \lor y \\
&v_3 \equiv \neg y \lor z
\end{aligned}$$

Tseitin's Encoding (example)

$$(x \to y) \land (y \to z) \to (x \to z)$$

- rewrite as a conjunction of definitions of new variables
- rewrite definitions only using $\neg$, $\land$, and $\lor$
- rewrite as a conjunction of clauses

$$\begin{array}{ll}
v_0 & v_0 \\
v_0 \equiv \neg v_1 \lor v_2 & (v_0 \to \neg v_1 \lor v_2) \land (\neg v_1 \lor v_2 \to v_0) \\
v_1 \equiv v_3 \land v_4 & (v_1 \to v_3 \land v_4) \land (v_3 \land v_4 \to v_1) \\
v_2 \equiv \neg x \lor z & (v_2 \to \neg x \lor z) \land (\neg x \lor z \to v_2) \\
v_3 \equiv \neg x \lor y & (v_3 \to \neg x \lor y) \land (\neg x \lor y \to v_3) \\
v_3 \equiv \neg y \lor z & (v_3 \to \neg y \lor z) \land (\neg y \lor z \to v_3)
\end{array}$$

Tseitin's Encoding (example)

$$(x \to y) \wedge (y \to z) \to (x \to z)$$

- rewrite as a conjunction of definitions of new variables
- rewrite definitions only using $\neg$, $\wedge$, and $\vee$
- rewrite as a conjunction of clauses

v_0	v_0
$v_0 \equiv \neg v_1 \vee v_2$	$(v_0 \to \neg v_1 \vee v_2) \wedge (\neg v_1 \to v_0) \wedge (v_2 \to v_0)$
$v_1 \equiv v_3 \wedge v_4$	$(v_1 \to v_3) \wedge (v_1 \to v4) \wedge (v_3 \wedge v_4 \to v_1)$
$v_2 \equiv \neg x \vee z$	$(v_2 \to \neg x \vee z) \wedge (\neg x \to v_2) \wedge (z \to v_2)$
$v_3 \equiv \neg x \vee y$	$(v_3 \to \neg x \vee y) \wedge (\neg x \to v_3) \wedge (y \to v_3)$
$v_3 \equiv \neg y \vee z$	$(v_3 \to \neg y \vee z) \wedge (\neg y \to v_3) \wedge (z \to v_3)$

Tseitin's Encoding (example)

$$(x \rightarrow y) \wedge (y \rightarrow z) \rightarrow (x \rightarrow z)$$

- rewrite as a conjunction of definitions of new variables
- rewrite definitions only using $\neg$, $\wedge$, and $\vee$
- rewrite as a conjunction of clauses

v_0	v_0
$v_0 \equiv \neg v_1 \vee v_2$	$(\neg v_0 \vee \neg v_1 \vee v_2) \wedge (v_1 \vee v_0) \wedge (\neg v_2 \vee v_0)$
$v_1 \equiv v_3 \wedge v_4$	$(\neg v_1 \vee v_3) \wedge (\neg v_1 \vee v_4) \wedge (\neg v_3 \vee \neg v_4 \vee v_1)$
$v_2 \equiv \neg x \vee z$	$(\neg v_2 \vee \neg x \vee z) \wedge (x \vee v_2) \wedge (\neg z \vee v_2)$
$v_3 \equiv \neg x \vee y$	$(\neg v_3 \vee \neg x \vee y) \wedge (x \vee v_3) \wedge (\neg y \vee v_3)$
$v_3 \equiv \neg y \vee z$	$(\neg v_3 \vee \neg y \vee z) \wedge (y \vee v_3) \wedge (\neg z \vee v_3)$

Tseitin's Encoding (example)

$$(x \rightarrow y) \wedge (y \rightarrow z) \rightarrow (x \rightarrow z)$$

- rewrite as a conjunction of definitions of new variables
- rewrite definitions only using $\neg$, $\wedge$, and $\vee$
- rewrite as a conjunction of clauses

$$\begin{array}{ll}
v_0 & v_0 \\
v_0 \equiv \neg v_1 \vee v_2 & (\neg v_0 \vee \neg v_1 \vee v_2) \wedge (v_1 \vee v_0) \wedge (\neg v_2 \vee v_0) \\
v_1 \equiv v_3 \wedge v_4 & (\neg v_1 \vee v_3) \wedge (\neg v_1 \vee v_4) \wedge (\neg v_3 \vee \neg v_4 \vee v_1) \\
v_2 \equiv \neg x \vee z & (\neg v_2 \vee \neg x \vee z) \wedge (x \vee v_2) \wedge (\neg z \vee v_2) \\
v_3 \equiv \neg x \vee y & (\neg v_3 \vee \neg x \vee y) \wedge (x \vee v_3) \wedge (\neg y \vee v_3) \\
v_3 \equiv \neg y \vee z & (\neg v_3 \vee \neg y \vee z) \wedge (y \vee v_3) \wedge (\neg z \vee v_3)
\end{array}$$

The CNF we obtain is equisatisfiable, but not equivalent.

Certifying Unsatisfiability

- If a CNF φ is satisfiable, then a SAT solver returns a model
  - it is easy to check its correctness
- If φ is unsatisfiable and a SAT solver says so, how to check that the answer is correct?
- SAT solvers can return the proof of unsatisfiability
- Usually, some representation of a resolution refutation of φ

Resolution

Given clauses A, B, and a variable x

$$\frac{A \vee x \qquad B \vee \neg x}{A \vee B}$$

- $A \vee B$ is the resolvent of parent clauses $A \vee x$ and $B \vee \neg x$.

Definition

Resolution derivation of a clause C from a CNF φ is a sequence of clauses $C_1, \ldots, C_k$ such that $C_k = C$ and for each $i = 1, \ldots, k$ either

- $C_i \in \varphi$, or
- C_i is a resolvent of some clauses preceding it in the list.

Resolution refutation of φ is the resolution derivation of the empty clause $\perp$ (contradiction).

Resolution Graph

- Resolution derivation can be represented with a resolution graph
- It is a directed acyclic graph with

nodes	clauses
leaves	clauses of φ
inner nodes	resolvents
edges	from parent clauses to resolvents
sink node	the derived clause ($\perp$ in case of a refutation)

Resolution Graph (example)

$$\varphi = \neg x \wedge (y \vee \neg z) \wedge (x \vee \neg y) \wedge (x \vee y \vee z) \wedge (\neg x \vee z)$$

$$\begin{tikzcd}
& y \vee \neg z \arrow[dr] & & x \vee \neg y \arrow[dl] \arrow[dr] & & x \vee y \vee z \arrow[dl] & \\
\neg x \arrow[drr] & & x \vee \neg z \arrow[d] & & x \vee z \arrow[d] & & \neg x \vee z \arrow[dll] \\
& & \neg z \arrow[dr] & & z \arrow[dl] & & \\
& & & \bot & & &
\end{tikzcd}$$

Tree Resolution

Tree resolution refutation — the resolution graph is a tree (clauses from φ can be in several leaves)

$$\varphi = \neg x \wedge (y \vee \neg z) \wedge (x \vee \neg y) \wedge (x \vee y \vee z) \wedge (\neg x \vee z)$$

$$\begin{tikzcd}
& y \vee \neg z \arrow[dr] & & x \vee \neg y \arrow[dl] & & x \vee \neg y \arrow[dr] & & x \vee y \vee z \arrow[dl] & \\
\neg x \arrow[dr] & & x \vee \neg z \arrow[dl] & & & & x \vee z \arrow[dr] & & \neg x \vee z \arrow[dl] \\
& \neg z \arrow[drrr] & & & & & & z \arrow[dlll] & \\
& & & & \perp & & & &
\end{tikzcd}$$

Resolution and Tree Resolution

$\varphi \models C$ every model of φ is a model of C as well.

- Clause C is an implicate of φ
- Equivalent to $\varphi \wedge \bigwedge_{l \in C} \neg l \models \bot$

$\varphi \models \bot$ if and only if φ is unsatisfiable

$\varphi \vdash C$ clause C can be derived by resolution from φ.

- Resolution is sound and complete, for every CNF formula φ

$$\varphi \models \bot \text{ if and only if } \varphi \vdash \bot$$

- Resolution refutations can have exponential length
  - Pigeon hole principle formulas.
- Tree resolution is sound and complete
- Tree resolution refutations can be exponentially longer than general resolution refutations.

Unit Resolution

unit resolution one of the parent clauses is a unit clause (single literal)

$\varphi \vdash_1 C$ clause C can be derived from φ by unit resolution

- Unit resolution is sound but incomplete
- For some unsatisfiable formulas, contradiction cannot be derived by unit resolution
- Unit resolution refutation can be found in linear time if it exists

Unit Resolution (Example)

$$\varphi = \neg x \wedge (y \vee \neg z) \wedge (x \vee \neg y) \wedge (x \vee y \vee z) \wedge (\neg x \vee z)$$

$$\begin{tikzcd}
 & x \vee \neg y \arrow[dr] & & \neg x \arrow[dl] \arrow[dr] & & x \vee y \vee z \arrow[dl] \\
y \vee \neg z \arrow[dr] & & \neg y \arrow[dl] \arrow[dr] & & y \vee z \arrow[dl] & \\
 & \neg z \arrow[dr] & & z \arrow[dl] & & \\
 & & \perp & & &
\end{tikzcd}$$

Partial Assignments — Notation

$\mathbf{x}$ set of variables

$\mathrm{lit}(\mathbf{x})$ literals over variables in $\mathbf{x}$

partial assignment a non-contradictory set of literals, considered as a conjunction of literals

$\varphi[\alpha]$ Application of a partial assignment $\alpha \subseteq \mathrm{lit}(\mathbf{x})$:

- Clauses containing a literal from α are removed from φ and
- negations of literals in α are removed from the remaining clauses.

$$\varphi = (y \vee \neg z) \wedge (x \vee \neg y) \wedge (x \vee y \vee z) \wedge (\neg x \vee z)$$

$$\varphi[\neg x] = (y \vee \neg z) \wedge (\neg y) \wedge (y \vee z)$$

$\varphi[x, \neg z] = \bot$ (The empty clause — contradiction)

$\varphi[x, y, z] = \top$ (The empty CNF — satisfied)

Unit Propagation Algorithm

Function `UnitProp(`φ`)`

Input: CNF formula φ on variables $\mathbf{x}$

Output: (α, ψ) where α is a set of literals which can be derived by unit resolution from φ, $\psi = \varphi[\alpha]$.

$\alpha \leftarrow \emptyset$

while φ contains a unit clause l **do**

  $\alpha \leftarrow \alpha \cup \{l\}$

  $\varphi \leftarrow \varphi[l]$

  if $\bot \in \varphi$ **then return** $(\alpha, \bot)$

end

return (α, φ)

- Efficent procedure
  - linear time implementation
  - Efficient data structures (watched literals)
- Used very often in SAT solvers

Unit Propagation Example

1. $\varphi = \underbrace{\neg x}_{\text{unit clause}} \wedge (y \vee \neg z) \wedge (x \vee \neg y) \wedge (x \vee y \vee z) \wedge (\neg x \vee z)$, $\alpha = \emptyset$

2. $\varphi = \neg x \wedge (y \vee \neg z) \wedge \underbrace{(x \vee \neg y)}_{\text{unit clause}} \wedge (x \vee y \vee z) \wedge (\neg x \vee z)$, $\alpha = \{\neg x\}$

3. $\varphi = \neg x \wedge (y \vee \neg z) \wedge (x \vee \neg y) \wedge \underbrace{(x \vee y \vee z)}_{\text{unit clause}} \wedge (\neg x \vee z)$,

   $\alpha = \{\neg x, \neg y\}$

4. $\varphi = \neg x \wedge \underbrace{(y \vee \neg z)}_{\text{empty clause}} \wedge (x \vee \neg y) \wedge (x \vee y \vee z) \wedge (\neg x \vee z)$,

   $\alpha = \{\neg x, \neg y, z\}$

Unit Propagation Example

1. $\varphi = \underbrace{\neg x}_{\text{unit clause}} \wedge (y \vee \neg z) \wedge (x \vee \neg y) \wedge (x \vee y \vee z) \wedge (\neg x \vee z)$, $\alpha = \emptyset$

2. $\varphi = \neg x \wedge (y \vee \neg z) \wedge \underbrace{(x \vee \neg y)}_{\text{unit clause}} \wedge (x \vee y \vee z) \wedge (\neg x \vee z)$, $\alpha = \{\neg x\}$

3. $\varphi = \neg x \wedge (y \vee \neg z) \wedge (x \vee \neg y) \wedge \underbrace{(x \vee y \vee z)}_{\text{unit clause}} \wedge (\neg x \vee z)$,

   $\alpha = \{\neg x, \neg y\}$

4. $\varphi = \neg x \wedge \underbrace{(y \vee \neg z)}_{\text{empty clause}} \wedge (x \vee \neg y) \wedge (x \vee y \vee z) \wedge (\neg x \vee z)$,

   $\alpha = \{\neg x, \neg y, z\}$

Unit Propagation Example

1. $\varphi = \underbrace{\neg x}_{\text{unit clause}} \wedge (y \vee \neg z) \wedge (x \vee \neg y) \wedge (x \vee y \vee z) \wedge (\neg x \vee z)$, $\alpha = \emptyset$

2. $\varphi = \neg x \wedge (y \vee \neg z) \wedge \underbrace{(x \vee \neg y)}_{\text{unit clause}} \wedge (x \vee y \vee z) \wedge (\neg x \vee z)$, $\alpha = \{\neg x\}$

3. $\varphi = \neg x \wedge (y \vee \neg z) \wedge (x \vee \neg y) \wedge \underbrace{(x \vee y \vee z)}_{\text{unit clause}} \wedge (\neg x \vee z)$,

   $\alpha = \{\neg x, \neg y\}$

4. $\varphi = \neg x \wedge \underbrace{(y \vee \neg z)}_{\text{empty clause}} \wedge (x \vee \neg y) \wedge (x \vee y \vee z) \wedge (\neg x \vee z)$,

   $\alpha = \{\neg x, \neg y, z\}$

Unit Propagation Example

1. $\varphi = \underbrace{\neg x}_{\text{unit clause}} \wedge (y \vee \neg z) \wedge (x \vee \neg y) \wedge (x \vee y \vee z) \wedge (\neg x \vee z)$, $\alpha = \emptyset$

2. $\varphi = \neg x \wedge (y \vee \neg z) \wedge \underbrace{(x \vee \neg y)}_{\text{unit clause}} \wedge (x \vee y \vee z) \wedge (\neg x \vee z)$, $\alpha = \{\neg x\}$

3. $\varphi = \neg x \wedge (y \vee \neg z) \wedge (x \vee \neg y) \wedge \underbrace{(x \vee y \vee z)}_{\text{unit clause}} \wedge (\neg x \vee z)$,

   $\alpha = \{\neg x, \neg y\}$

4. $\varphi = \neg x \wedge \underbrace{(y \vee \neg z)}_{\text{empty clause}} \wedge (x \vee \neg y) \wedge (x \vee y \vee z) \wedge (\neg x \vee z)$,

   $\alpha = \{\neg x, \neg y, z\}$

Unit Propagation Example

1. $\varphi = \underbrace{\neg x}_{\text{unit clause}} \wedge (y \vee \neg z) \wedge (x \vee \neg y) \wedge (x \vee y \vee z) \wedge (\neg x \vee z)$, $\alpha = \emptyset$

2. $\varphi = \neg x \wedge (y \vee \neg z) \wedge \underbrace{(x \vee \neg y)}_{\text{unit clause}} \wedge (x \vee y \vee z) \wedge (\neg x \vee z)$, $\alpha = \{\neg x\}$

3. $\varphi = \neg x \wedge (y \vee \neg z) \wedge (x \vee \neg y) \wedge \underbrace{(x \vee y \vee z)}_{\text{unit clause}} \wedge (\neg x \vee z)$,

   $\alpha = \{\neg x, \neg y\}$

4. $\varphi = \neg x \wedge \underbrace{(y \vee \neg z)}_{\text{empty clause}} \wedge (x \vee \neg y) \wedge (x \vee y \vee z) \wedge (\neg x \vee z)$,

   $\alpha = \{\neg x, \neg y, z\}$

Empty clause derived — Unit propagation returns ($\{\neg x, \neg y, z\}$, $\bot$).

DPLL

DPLL

- DP algorithm — existential quantification (Davis and Putnam, 1960) by DP-elimination
  - High space complexity (can soon blow up exponentially)
- Davis Putnam Logemann Loveland (Davis, Logemann, and Loveland, 1962)
- Branch and bound algorithm
  - Branch on values of a variable
  - Use unit propagation to prune the search tree
  - Polynomial space complexity

DPLL Algorithm

Function `DPLL`(CNF φ)

Output: A set of literals (partial model) or `UNSAT`

$(\alpha, \psi) \leftarrow \texttt{UnitProp}(\varphi)$
if $\psi = \emptyset$ **then return** α
if $\bot \in \psi$ **then return** `UNSAT`
$l \leftarrow$ a literal in ψ
$\beta \leftarrow \texttt{DPLL}(\psi[l])$
if $\beta \neq \texttt{UNSAT}$ **then return** $\alpha \cup \beta \cup \{l\}$
$\beta \leftarrow \texttt{DPLL}(\psi[\neg l])$
if $\beta \neq \texttt{UNSAT}$ **then return** $\alpha \cup \beta \cup \{\neg l\}$
return `UNSAT`

DPLL — running example

$$\varphi = (x_1 \vee x_3 \vee x_4) \wedge (\neg x_1 \vee x_2 \vee x_3) \wedge (\neg x_1 \vee \neg x_3) \wedge (\neg x_2 \vee \neg x_4) \wedge (x_3 \vee x_4)$$

1. Decide x_1:
$\underbrace{(x_1 \vee x_3 \vee x_4) \wedge (\neg x_1 \vee x_2 \vee x_3) \wedge (\neg x_1 \vee \neg x_3)}_{\text{unit clause}} \wedge (\neg x_2 \vee \neg x_4) \wedge (x_3 \vee x_4)$

2. Derive $\neg x_3$ by unit propagation:
$(x_1 \vee x_3 \vee x_4) \wedge \underbrace{(\neg x_1 \vee x_2 \vee x_3)}_{\text{unit clause}} \wedge (\neg x_1 \vee \neg x_3) \wedge (\neg x_2 \vee \neg x_4) \wedge \underbrace{(x_3 \vee x_4)}_{\text{unit clause}}$

3. Derive x_2, x_4 by unit propagation:
$(x_1 \vee x_3 \vee x_4) \wedge (\neg x_1 \vee x_2 \vee x_3) \wedge (\neg x_1 \vee \neg x_3) \wedge \underbrace{(\neg x_2 \vee \neg x_4)}_{\text{empty clause}} \wedge (x_3 \vee x_4)$

1. Decide $\neg x_1$:
$(x_1 \vee x_3 \vee x_4) \wedge (\neg x_1 \vee x_2 \vee x_3) \wedge (\neg x_1 \vee \neg x_3) \wedge (\neg x_2 \vee \neg x_4) \wedge (x_3 \vee x_4)$

2. Decide x_2:
$(x_1 \vee x_3 \vee x_4) \wedge (\neg x_1 \vee x_2 \vee x_3) \wedge (\neg x_1 \vee \neg x_3) \wedge \underbrace{(\neg x_2 \vee \neg x_4)}_{\text{unit clause}} \wedge (x_3 \vee x_4)$

3. Derive $\neg x_4$ by unit propagation:
$(x_1 \vee x_3 \vee x_4) \wedge (\neg x_1 \vee x_2 \vee x_3) \wedge (\neg x_1 \vee \neg x_3) \wedge (\neg x_2 \vee \neg x_4) \wedge \underbrace{(x_3 \vee x_4)}_{\text{unit clause}}$

4. Derive x_3 by unit propagation:
$(x_1 \vee x_3 \vee x_4) \wedge (\neg x_1 \vee x_2 \vee x_3) \wedge (\neg x_1 \vee \neg x_3) \wedge (\neg x_2 \vee \neg x_4) \wedge (x_3 \vee x_4)$

DPLL — running example

$$\varphi = (x_1 \vee x_3 \vee x_4) \wedge (\neg x_1 \vee x_2 \vee x_3) \wedge (\neg x_1 \vee \neg x_3) \wedge (\neg x_2 \vee \neg x_4) \wedge (x_3 \vee x_4)$$

1. Decide x_1:
$(x_1 \vee x_3 \vee x_4) \wedge (\neg x_1 \vee x_2 \vee x_3) \wedge \underbrace{(\neg x_1 \vee \neg x_3)}_{\text{unit clause}} \wedge (\neg x_2 \vee \neg x_4) \wedge (x_3 \vee x_4)$

2. Derive $\neg x_3$ by unit propagation:
$(x_1 \vee x_3 \vee x_4) \wedge \underbrace{(\neg x_1 \vee x_2 \vee x_3)}_{\text{unit clause}} \wedge (\neg x_1 \vee \neg x_3) \wedge (\neg x_2 \vee \neg x_4) \wedge \underbrace{(x_3 \vee x_4)}_{\text{unit clause}}$

3. Derive x_2, x_4 by unit propagation:
$(x_1 \vee x_3 \vee x_4) \wedge (\neg x_1 \vee x_2 \vee x_3) \wedge (\neg x_1 \vee \neg x_3) \wedge \underbrace{(\neg x_2 \vee \neg x_4)}_{\text{empty clause}} \wedge (x_3 \vee x_4)$

1. Decide $\neg x_1$:
$(x_1 \vee x_3 \vee x_4) \wedge (\neg x_1 \vee x_2 \vee x_3) \wedge (\neg x_1 \vee \neg x_3) \wedge (\neg x_2 \vee \neg x_4) \wedge (x_3 \vee x_4)$

2. Decide x_2:
$(x_1 \vee x_3 \vee x_4) \wedge (\neg x_1 \vee x_2 \vee x_3) \wedge (\neg x_1 \vee \neg x_3) \wedge \underbrace{(\neg x_2 \vee \neg x_4)}_{\text{unit clause}} \wedge (x_3 \vee x_4)$

3. Derive $\neg x_4$ by unit propagation:
$(x_1 \vee x_3 \vee x_4) \wedge (\neg x_1 \vee x_2 \vee x_3) \wedge (\neg x_1 \vee \neg x_3) \wedge (\neg x_2 \vee \neg x_4) \wedge \underbrace{(x_3 \vee x_4)}_{\text{unit clause}}$

4. Derive x_3 by unit propagation:
$(x_1 \vee x_3 \vee x_4) \wedge (\neg x_1 \vee x_2 \vee x_3) \wedge (\neg x_1 \vee \neg x_3) \wedge (\neg x_2 \vee \neg x_4) \wedge (x_3 \vee x_4)$

DPLL — running example

$$\varphi = (x_1 \lor x_3 \lor x_4) \land (\neg x_1 \lor x_2 \lor x_3) \land (\neg x_1 \lor \neg x_3) \land (\neg x_2 \lor \neg x_4) \land (x_3 \lor x_4)$$

1. Decide x_1:
   $(x_1 \lor x_3 \lor x_4) \land (\neg x_1 \lor x_2 \lor x_3) \land \underbrace{(\neg x_1 \lor \neg x_3)}_{\text{unit clause}} \land (\neg x_2 \lor \neg x_4) \land (x_3 \lor x_4)$
2. Derive $\neg x_3$ by unit propagation:
   $(x_1 \lor x_3 \lor x_4) \land \underbrace{(\neg x_1 \lor x_2 \lor x_3)}_{\text{unit clause}} \land (\neg x_1 \lor \neg x_3) \land (\neg x_2 \lor \neg x_4) \land \underbrace{(x_3 \lor x_4)}_{\text{unit clause}}$
3. Derive x_2, x_4 by unit propagation:
   $(x_1 \lor x_3 \lor x_4) \land (\neg x_1 \lor x_2 \lor x_3) \land (\neg x_1 \lor \neg x_3) \land \underbrace{(\neg x_2 \lor \neg x_4)}_{\text{empty clause}} \land (x_3 \lor x_4)$

1. Decide $\neg x_1$:
   $(x_1 \lor x_3 \lor x_4) \land (\neg x_1 \lor x_2 \lor x_3) \land (\neg x_1 \lor \neg x_3) \land (\neg x_2 \lor \neg x_4) \land (x_3 \lor x_4)$
2. Decide x_2:
   $(x_1 \lor x_3 \lor x_4) \land (\neg x_1 \lor x_2 \lor x_3) \land (\neg x_1 \lor \neg x_3) \land \underbrace{(\neg x_2 \lor \neg x_4)}_{\text{unit clause}} \land (x_3 \lor x_4)$
3. Derive $\neg x_4$ by unit propagation:
   $(x_1 \lor x_3 \lor x_4) \land (\neg x_1 \lor x_2 \lor x_3) \land (\neg x_1 \lor \neg x_3) \land (\neg x_2 \lor \neg x_4) \land \underbrace{(x_3 \lor x_4)}_{\text{unit clause}}$
4. Derive x_3 by unit propagation:
   $(x_1 \lor x_3 \lor x_4) \land (\neg x_1 \lor x_2 \lor x_3) \land (\neg x_1 \lor \neg x_3) \land (\neg x_2 \lor \neg x_4) \land (x_3 \lor x_4)$

DPLL — running example

$$\varphi = (x_1 \vee x_3 \vee x_4) \wedge (\neg x_1 \vee x_2 \vee x_3) \wedge (\neg x_1 \vee \neg x_3) \wedge (\neg x_2 \vee \neg x_4) \wedge (x_3 \vee x_4)$$

1. Decide x_1:
   $$(x_1 \vee x_3 \vee x_4) \wedge (\neg x_1 \vee x_2 \vee x_3) \wedge \underbrace{(\neg x_1 \vee \neg x_3)}_{\text{unit clause}} \wedge (\neg x_2 \vee \neg x_4) \wedge (x_3 \vee x_4)$$
2. Derive $\neg x_3$ by unit propagation:
   $$(x_1 \vee x_3 \vee x_4) \wedge \underbrace{(\neg x_1 \vee x_2 \vee x_3)}_{\text{unit clause}} \wedge (\neg x_1 \vee \neg x_3) \wedge (\neg x_2 \vee \neg x_4) \wedge \underbrace{(x_3 \vee x_4)}_{\text{unit clause}}$$
3. Derive x_2, x_4 by unit propagation:
   $$(x_1 \vee x_3 \vee x_4) \wedge (\neg x_1 \vee x_2 \vee x_3) \wedge (\neg x_1 \vee \neg x_3) \wedge \underbrace{(\neg x_2 \vee \neg x_4)}_{\text{empty clause}} \wedge (x_3 \vee x_4)$$

1. Decide $\neg x_1$:
   $$(x_1 \vee x_3 \vee x_4) \wedge (\neg x_1 \vee x_2 \vee x_3) \wedge (\neg x_1 \vee \neg x_3) \wedge (\neg x_2 \vee \neg x_4) \wedge (x_3 \vee x_4)$$
2. Decide x_2:
   $$(x_1 \vee x_3 \vee x_4) \wedge (\neg x_1 \vee x_2 \vee x_3) \wedge (\neg x_1 \vee \neg x_3) \wedge \underbrace{(\neg x_2 \vee \neg x_4)}_{\text{unit clause}} \wedge (x_3 \vee x_4)$$
3. Derive $\neg x_4$ by unit propagation:
   $$(x_1 \vee x_3 \vee x_4) \wedge (\neg x_1 \vee x_2 \vee x_3) \wedge (\neg x_1 \vee \neg x_3) \wedge (\neg x_2 \vee \neg x_4) \wedge \underbrace{(x_3 \vee x_4)}_{\text{unit clause}}$$
4. Derive x_3 by unit propagation:
   $$(x_1 \vee x_3 \vee x_4) \wedge (\neg x_1 \vee x_2 \vee x_3) \wedge (\neg x_1 \vee \neg x_3) \wedge (\neg x_2 \vee \neg x_4) \wedge (x_3 \vee x_4)$$

DPLL — running example

$$\varphi = (x_1 \vee x_3 \vee x_4) \wedge (\neg x_1 \vee x_2 \vee x_3) \wedge (\neg x_1 \vee \neg x_3) \wedge (\neg x_2 \vee \neg x_4) \wedge (x_3 \vee x_4)$$

1. Decide x_1:
$(x_1 \vee x_3 \vee x_4) \wedge (\neg x_1 \vee x_2 \vee x_3) \wedge \underbrace{(\neg x_1 \vee \neg x_3)}_{\text{unit clause}} \wedge (\neg x_2 \vee \neg x_4) \wedge (x_3 \vee x_4)$

2. Derive $\neg x_3$ by unit propagation:
$(x_1 \vee x_3 \vee x_4) \wedge \underbrace{(\neg x_1 \vee x_2 \vee x_3)}_{\text{unit clause}} \wedge (\neg x_1 \vee \neg x_3) \wedge (\neg x_2 \vee \neg x_4) \wedge \underbrace{(x_3 \vee x_4)}_{\text{unit clause}}$

3. Derive x_2, x_4 by unit propagation:
$(x_1 \vee x_3 \vee x_4) \wedge (\neg x_1 \vee x_2 \vee x_3) \wedge (\neg x_1 \vee \neg x_3) \wedge \underbrace{(\neg x_2 \vee \neg x_4)}_{\text{empty clause}} \wedge (x_3 \vee x_4)$

1. Decide $\neg x_1$:
$(x_1 \vee x_3 \vee x_4) \wedge (\neg x_1 \vee x_2 \vee x_3) \wedge (\neg x_1 \vee \neg x_3) \wedge (\neg x_2 \vee \neg x_4) \wedge (x_3 \vee x_4)$

2. Decide x_2:
$(x_1 \vee x_3 \vee x_4) \wedge (\neg x_1 \vee x_2 \vee x_3) \wedge (\neg x_1 \vee \neg x_3) \wedge \underbrace{(\neg x_2 \vee \neg x_4)}_{\text{unit clause}} \wedge (x_3 \vee x_4)$

3. Derive $\neg x_4$ by unit propagation:
$(x_1 \vee x_3 \vee x_4) \wedge (\neg x_1 \vee x_2 \vee x_3) \wedge (\neg x_1 \vee \neg x_3) \wedge (\neg x_2 \vee \neg x_4) \wedge \underbrace{(x_3 \vee x_4)}_{\text{unit clause}}$

4. Derive x_3 by unit propagation:
$(x_1 \vee x_3 \vee x_4) \wedge (\neg x_1 \vee x_2 \vee x_3) \wedge (\neg x_1 \vee \neg x_3) \wedge (\neg x_2 \vee \neg x_4) \wedge (x_3 \vee x_4)$

DPLL — running example

$$\varphi = (x_1 \vee x_3 \vee x_4) \wedge (\neg x_1 \vee x_2 \vee x_3) \wedge (\neg x_1 \vee \neg x_3) \wedge (\neg x_2 \vee \neg x_4) \wedge (x_3 \vee x_4)$$

1. Decide x_1:
$(x_1 \vee x_3 \vee x_4) \wedge (\neg x_1 \vee x_2 \vee x_3) \wedge$
$\underbrace{(\neg x_1 \vee \neg x_3)}_{\text{unit clause}} \wedge (\neg x_2 \vee \neg x_4) \wedge (x_3 \vee x_4)$

2. Derive $\neg x_3$ by unit propagation:
$(x_1 \vee x_3 \vee x_4) \wedge \underbrace{(\neg x_1 \vee x_2 \vee x_3)}_{\text{unit clause}} \wedge (\neg x_1 \vee$
$\neg x_3) \wedge (\neg x_2 \vee \neg x_4) \wedge \underbrace{(x_3 \vee x_4)}_{\text{unit clause}}$

3. Derive x_2, x_4 by unit propagation:
$(x_1 \vee x_3 \vee x_4) \wedge (\neg x_1 \vee x_2 \vee x_3) \wedge$
$(\neg x_1 \vee \neg x_3) \wedge \underbrace{(\neg x_2 \vee \neg x_4)}_{\text{empty clause}} \wedge (x_3 \vee x_4)$

1. Decide $\neg x_1$:
$(x_1 \vee x_3 \vee x_4) \wedge (\neg x_1 \vee x_2 \vee x_3) \wedge$
$(\neg x_1 \vee \neg x_3) \wedge (\neg x_2 \vee \neg x_4) \wedge (x_3 \vee x_4)$

2. Decide x_2:
$(x_1 \vee x_3 \vee x_4) \wedge (\neg x_1 \vee x_2 \vee x_3) \wedge$
$(\neg x_1 \vee \neg x_3) \wedge \underbrace{(\neg x_2 \vee \neg x_4)}_{\text{unit clause}} \wedge (x_3 \vee x_4)$

3. Derive $\neg x_4$ by unit propagation:
$(x_1 \vee x_3 \vee x_4) \wedge (\neg x_1 \vee x_2 \vee x_3) \wedge$
$(\neg x_1 \vee \neg x_3) \wedge (\neg x_2 \vee \neg x_4) \wedge \underbrace{(x_3 \vee x_4)}_{\text{unit clause}}$

4. Derive x_3 by unit propagation:
$(x_1 \vee x_3 \vee x_4) \wedge (\neg x_1 \vee x_2 \vee x_3) \wedge$
$(\neg x_1 \vee \neg x_3) \wedge (\neg x_2 \vee \neg x_4) \wedge (x_3 \vee x_4)$

DPLL — running example

$$\varphi = (x_1 \vee x_3 \vee x_4) \wedge (\neg x_1 \vee x_2 \vee x_3) \wedge (\neg x_1 \vee \neg x_3) \wedge (\neg x_2 \vee \neg x_4) \wedge (x_3 \vee x_4)$$

1. Decide x_1:
   $(x_1 \vee x_3 \vee x_4) \wedge (\neg x_1 \vee x_2 \vee x_3) \wedge \underbrace{(\neg x_1 \vee \neg x_3)}_{\text{unit clause}} \wedge (\neg x_2 \vee \neg x_4) \wedge (x_3 \vee x_4)$
2. Derive $\neg x_3$ by unit propagation:
   $(x_1 \vee x_3 \vee x_4) \wedge \underbrace{(\neg x_1 \vee x_2 \vee x_3)}_{\text{unit clause}} \wedge (\neg x_1 \vee \neg x_3) \wedge (\neg x_2 \vee \neg x_4) \wedge \underbrace{(x_3 \vee x_4)}_{\text{unit clause}}$
3. Derive x_2, x_4 by unit propagation:
   $(x_1 \vee x_3 \vee x_4) \wedge (\neg x_1 \vee x_2 \vee x_3) \wedge (\neg x_1 \vee \neg x_3) \wedge \underbrace{(\neg x_2 \vee \neg x_4)}_{\text{empty clause}} \wedge (x_3 \vee x_4)$

1. Decide $\neg x_1$:
   $(x_1 \vee x_3 \vee x_4) \wedge (\neg x_1 \vee x_2 \vee x_3) \wedge (\neg x_1 \vee \neg x_3) \wedge (\neg x_2 \vee \neg x_4) \wedge (x_3 \vee x_4)$
2. Decide x_2:
   $(x_1 \vee x_3 \vee x_4) \wedge (\neg x_1 \vee x_2 \vee x_3) \wedge (\neg x_1 \vee \neg x_3) \wedge \underbrace{(\neg x_2 \vee \neg x_4)}_{\text{unit clause}} \wedge (x_3 \vee x_4)$
3. Derive $\neg x_4$ by unit propagation:
   $(x_1 \vee x_3 \vee x_4) \wedge (\neg x_1 \vee x_2 \vee x_3) \wedge (\neg x_1 \vee \neg x_3) \wedge (\neg x_2 \vee \neg x_4) \wedge \underbrace{(x_3 \vee x_4)}_{\text{unit clause}}$
4. Derive x_3 by unit propagation:
   $(x_1 \vee x_3 \vee x_4) \wedge (\neg x_1 \vee x_2 \vee x_3) \wedge (\neg x_1 \vee \neg x_3) \wedge (\neg x_2 \vee \neg x_4) \wedge (x_3 \vee x_4)$

DPLL — running example

$$\varphi = (x_1 \lor x_3 \lor x_4) \land (\neg x_1 \lor x_2 \lor x_3) \land (\neg x_1 \lor \neg x_3) \land (\neg x_2 \lor \neg x_4) \land (x_3 \lor x_4)$$

1. Decide x_1:
$$(x_1 \lor x_3 \lor x_4) \land (\neg x_1 \lor x_2 \lor x_3) \land \underbrace{(\neg x_1 \lor \neg x_3)}_{\text{unit clause}} \land (\neg x_2 \lor \neg x_4) \land (x_3 \lor x_4)$$

2. Derive $\neg x_3$ by unit propagation:
$$(x_1 \lor x_3 \lor x_4) \land \underbrace{(\neg x_1 \lor x_2 \lor x_3)}_{\text{unit clause}} \land (\neg x_1 \lor \neg x_3) \land (\neg x_2 \lor \neg x_4) \land \underbrace{(x_3 \lor x_4)}_{\text{unit clause}}$$

3. Derive x_2, x_4 by unit propagation:
$$(x_1 \lor x_3 \lor x_4) \land (\neg x_1 \lor x_2 \lor x_3) \land (\neg x_1 \lor \neg x_3) \land \underbrace{(\neg x_2 \lor \neg x_4)}_{\text{empty clause}} \land (x_3 \lor x_4)$$

1. Decide $\neg x_1$:
$$(x_1 \lor x_3 \lor x_4) \land (\neg x_1 \lor x_2 \lor x_3) \land (\neg x_1 \lor \neg x_3) \land (\neg x_2 \lor \neg x_4) \land (x_3 \lor x_4)$$

2. Decide x_2:
$$(x_1 \lor x_3 \lor x_4) \land (\neg x_1 \lor x_2 \lor x_3) \land (\neg x_1 \lor \neg x_3) \land \underbrace{(\neg x_2 \lor \neg x_4)}_{\text{unit clause}} \land (x_3 \lor x_4)$$

3. Derive $\neg x_4$ by unit propagation:
$$(x_1 \lor x_3 \lor x_4) \land (\neg x_1 \lor x_2 \lor x_3) \land (\neg x_1 \lor \neg x_3) \land (\neg x_2 \lor \neg x_4) \land \underbrace{(x_3 \lor x_4)}_{\text{unit clause}}$$

4. Derive x_3 by unit propagation:
$$(x_1 \lor x_3 \lor x_4) \land (\neg x_1 \lor x_2 \lor x_3) \land (\neg x_1 \lor \neg x_3) \land (\neg x_2 \lor \neg x_4) \land (x_3 \lor x_4)$$

DPLL — running example

$$\varphi = (x_1 \vee x_3 \vee x_4) \wedge (\neg x_1 \vee x_2 \vee x_3) \wedge (\neg x_1 \vee \neg x_3) \wedge (\neg x_2 \vee \neg x_4) \wedge (x_3 \vee x_4)$$

1. Decide x_1:
$(x_1 \vee x_3 \vee x_4) \wedge (\neg x_1 \vee x_2 \vee x_3) \wedge \underbrace{(\neg x_1 \vee \neg x_3)}_{\text{unit clause}} \wedge (\neg x_2 \vee \neg x_4) \wedge (x_3 \vee x_4)$

2. Derive $\neg x_3$ by unit propagation:
$(x_1 \vee x_3 \vee x_4) \wedge \underbrace{(\neg x_1 \vee x_2 \vee x_3)}_{\text{unit clause}} \wedge (\neg x_1 \vee \neg x_3) \wedge (\neg x_2 \vee \neg x_4) \wedge \underbrace{(x_3 \vee x_4)}_{\text{unit clause}}$

3. Derive x_2, x_4 by unit propagation:
$(x_1 \vee x_3 \vee x_4) \wedge (\neg x_1 \vee x_2 \vee x_3) \wedge (\neg x_1 \vee \neg x_3) \wedge \underbrace{(\neg x_2 \vee \neg x_4)}_{\text{empty clause}} \wedge (x_3 \vee x_4)$

1. Decide $\neg x_1$:
$(x_1 \vee x_3 \vee x_4) \wedge (\neg x_1 \vee x_2 \vee x_3) \wedge (\neg x_1 \vee \neg x_3) \wedge (\neg x_2 \vee \neg x_4) \wedge (x_3 \vee x_4)$

2. Decide x_2:
$(x_1 \vee x_3 \vee x_4) \wedge (\neg x_1 \vee x_2 \vee x_3) \wedge (\neg x_1 \vee \neg x_3) \wedge \underbrace{(\neg x_2 \vee \neg x_4)}_{\text{unit clause}} \wedge (x_3 \vee x_4)$

3. Derive $\neg x_4$ by unit propagation:
$(x_1 \vee x_3 \vee x_4) \wedge (\neg x_1 \vee x_2 \vee x_3) \wedge (\neg x_1 \vee \neg x_3) \wedge (\neg x_2 \vee \neg x_4) \wedge \underbrace{(x_3 \vee x_4)}_{\text{unit clause}}$

4. Derive x_3 by unit propagation:
$(x_1 \vee x_3 \vee x_4) \wedge (\neg x_1 \vee x_2 \vee x_3) \wedge (\neg x_1 \vee \neg x_3) \wedge (\neg x_2 \vee \neg x_4) \wedge (x_3 \vee x_4)$

$\alpha = \{\neg x_1, x_2, x_3, \neg x_4\}$ is a satisfying assignment of φ

DPLL Variants and Extensions

- CDCL
  - Decide quickly
  - Quickly arrive at a conflict (empty clause)
  - Learn from conflicts
- Look-ahead solvers
  - Spend more time with decisions
  - Simplify formula between decision (e.g. eliminate pure literals)
- Cube and Conquer
  - Use look-ahead solver to split into subproblems
  - Solve the subproblems using a CDCL solver
- Model counters and enumerators
- `DPLL(T)` — SAT/SMT

References

References I

- Cook, Stephen A. (1971). "The complexity of theorem-proving procedures". In: STOC '71: Proceedings of the third annual ACM symposium on Theory Shaker Heights, Ohio, United States: ACM, pp. 151–158. DOI: http://doi.acm.org/10.1145/800157.805047.
- Davis, Martin, George Logemann, and Donald Loveland (July 1962). "A machine program for theorem-proving". In: Commun. ACM 5.7, pp. 394–397. ISSN: 0001-0782. DOI: 10.1145/368273.368557.
- Davis, Martin and Hilary Putnam (July 1960). "A Computing Procedure for Quantification Theory". In: J. ACM 7.3, pp. 201–215. ISSN: 0004-5411. DOI: 10.1145/321033.321034.
- Levin, L. A. (1973). "Universal'nye zadachi perebora". In: Probl. peredachi inform. 9.3, pp. 115–116.