DPLL and CDCL algorithms

Text: Heinz-Dieter Ebbinghaus, Jörg Flum, Wolfgang Thomas, *Mathematical logic*, Chapter XI.5; Daniel Kroening, Ofer Strichman, *Decision procedures*, Sec 2.2.2

Kamal Lodaya

Bharat Gyan Vigyan Samiti Karnataka, IISc

March 2025

Decision problems for FOL (Hilbert, Ackermann 1928)

Model checking: Given model M, S-formula B, does $M \models B$? Requires a finite domain $D = \{d_1, \ldots, d_m\}$ for finite input. Then interp'n I, ass't s are polynomial in m.

Satisfiability: Given formula B, is there a model M such that $M \models B$ holds?

Entscheidungsproblem: Given formula B, does $M \models B$ hold for every model M? Dual to satisfiability.

Model checking: Think of $\exists x B(x)$ as $\bigvee_{i=1}^{m} B(d_i)$. The formula size increases polynomially in m.

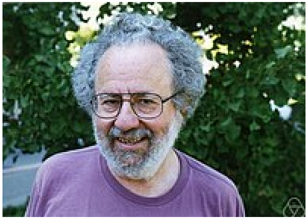
Parse tree for formula has nodes polynomial in size of the input. Use a traversal algorithm, this can be done in polynomial time.

Theorem (Turing 1936)

There is no algorithm for Entscheidungsproblem/FOL sat.

Sat solving (Martin Davis, Hilary Putnam 1960)

Fix a PL signature (propositional variables) Pr. Given formula B, is there an assignment s on Pr such that $s \models B$ holds?



Construct a truth table for B and check for evaluation to T. This can be done in time exponential in the number of variables in B.

How to do better?

Conjunctive normal form: Assume that formula B is in CNF, a set of clauses, written as $Th = \langle K_1; \ldots; K_m \rangle$. Each clause K_i is a set of literals, written as $[\ell_{i,1}, \ldots, \ell_{i,n}]$. (Note: A Horn clause is one with ≤ 1 positive literal.) Every clause is a consequence $Th \models K_i$.

Recall Resolution rule, preserving satisfiability. (Res): $\langle [B, A]; [\neg A, C] \rangle \implies [B, C]$.

Applying resolution on consequences gives a consequence.

Sat solving (Martin Davis, George Logemann and Donald Loveland 1962)

(UnitRes/One-Literal) If $[\ell]$ in clause set Th, delete clauses of Th containing ℓ (including unit clause) and occurrences in Th of complement $\bar{\ell}$. (No choice for satisfying assignment!)

(Affirmative-negative) If some literal ℓ occurs only positively/ only negatively in clause set Th, delete clauses in Th with ℓ.

(Split) Check sat of Th by testing sat of $\langle Th; [\ell]\rangle$ and of $\langle Th; [\bar{\ell}]\rangle$. $Th \models [\langle Th; [\ell]\rangle, \langle Th; [\bar{\ell}]\rangle]$, neither set may be a consequence.

DPLL ALGORITHM: First preprocess to CNF. Apply (UR) or (UR),(AN) eagerly (called Unit or Boolean constraint propagation). Otherwise choose a variable for (Split) and apply sequentially. If assignment to one value reaches a conflict, then backtrack to second value. Exponential when formula unsat.

D. Loveland

DPLL execution as search tree (Nordström 2022)

Example

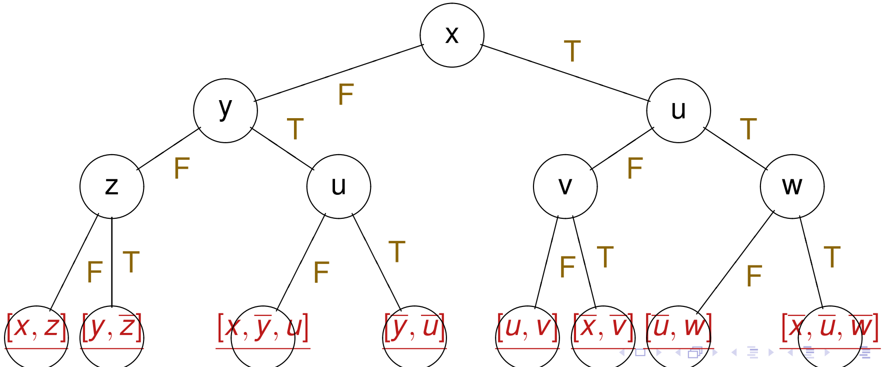
$B = (x \vee z) \wedge (z \rightarrow y) \wedge (y \rightarrow (x \vee u)) \wedge \neg(y \wedge u) \wedge (u \vee v) \wedge \neg(x \wedge v) \wedge (u \rightarrow w) \wedge \neg(x \wedge u \wedge w)$

gives $Th =$

$\langle [x, z]; [y, \overline{z}]; [x, \overline{y}, u]; [\overline{y}, \overline{u}]; [u, v]; [\overline{x}, \overline{v}]; [\overline{u}, w]; [\overline{x},$

Execution of DPLL (UP),(Split) visualized as a search tree.

G. Logemann

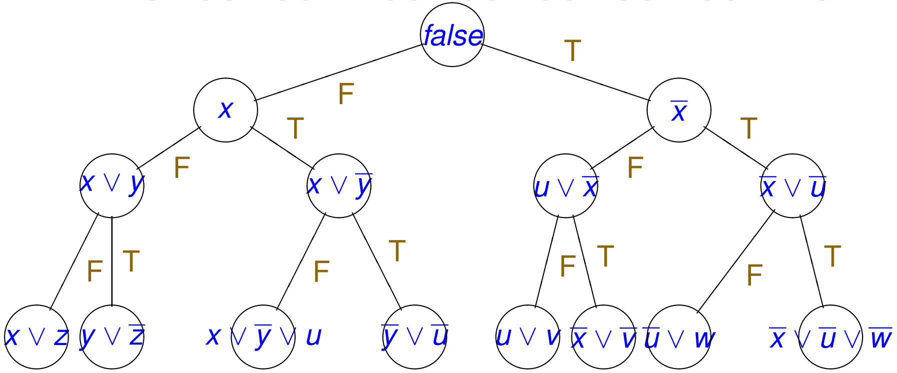

x
F
T
y
u
F
T
F
T
z
u
v
w
F
T
F
T
F
T
F
T
$[x, z]$
$[y, \overline{z}]$
$[x, \overline{y}, u]$
$[\overline{y}, \overline{u}]$
$[u, v]$
$[\overline{x}, \overline{v}]$
$[\overline{u}, w]$
$[\overline{x}, \overline{u}, \overline{w}]$

DPLL resolution tree (Nordström 2022)

Can visualize as Resolution calculus proof (Robinson 1968).

(Resolution) on pivot A: $\langle [B, A]; [\neg A, C] \rangle \implies [B, C]$ resolvent.

$Th = \langle [x, z]; [y, \overline{z}]; [x, \overline{y}, u]; [\overline{y}, \overline{u}]; [u, v]; [\overline{x}, \overline{v}]; [\overline{u}, w]; [\overline{x}, \overline{u}, \overline{w}] \rangle$



$false$
F
T
x
$\overline{x}$
F
T
F
T
$x \vee y$
$x \vee \overline{y}$
$u \vee \overline{x}$
$\overline{x} \vee \overline{u}$
F
T
F
T
F
T
F
T
$x \vee z$
$y \vee \overline{z}$
$x \vee \overline{y} \vee u$
$\overline{y} \vee \overline{u}$
$u \vee v$
$\overline{x} \vee \overline{v}$
$\overline{u} \vee w$
$\overline{x} \vee \overline{u} \vee \overline{w}$

Tree-like (resolution) proof: Proofs may not share subproofs.
DPLL resolution proofs are tree-like.

DPLL run: decision levels and trail

DL	*Partial assignment*	*Clauses*	*Trail*
0		$[\overline{p_1}, \overline{p_4}, p_3][\overline{p_3}, \overline{p_2}]$	
1	$p_1 \mapsto T$	$[\overline{p_1}, \overline{p_4}, p_3][\overline{p_3}, \overline{p_2}]$	$p_1@1$
2	$p_1 \mapsto T, p_4 \mapsto T$	$[\overline{p_1}, \overline{p_4}, p_3][\overline{p_3}, \overline{p_2}]$	$p_4@2$
UP	$p_1 \mapsto T, p_4 \mapsto T, p_3 \mapsto T$	$[p_3][\overline{p_3}, \overline{p_2}]$	$p_3@2$
UP	$p_1 \mapsto T, p_4 \mapsto T, p_3 \mapsto T, p_2 \mapsto F$	$[\overline{p_2}]$	$\overline{p_2}@2$

- Clause is satisfied if some literal evaluates to T under partial assignment
- Clause K is conflicting if all literals are assigned and K evaluates to F
- Clause K becomes unit if all but one literal is assigned and K is not satisfied
- Guessed assignments $p_1@1$ and $p_4@2$ are reasons of forced assignment $p_3@2$, which is reason of forced assignment $\overline{p_2}@2$.
  Clauses from which reasons come are antecedents.

DPLL run: guessed and forced assignments

Trail of splits (guessed/forced variables at decision levels):

DL	*Partial assignment*	*Clauses*	*Trail*
0		$[\overline{p_1}, \overline{p_4}, p_3][\overline{p_3}, \overline{p_2}]$	
1	$p_1 \mapsto T$	$[\overline{p_1}, \overline{p_4}, p_3][\overline{p_3}, \overline{p_2}]$	$p_1@1$
2	$p_1 \mapsto T, p_4 \mapsto T$	$[\overline{p_1}, \overline{p_4}, p_3][\overline{p_3}, \overline{p_2}]$	$p_4@2$
UP	$p_1 \mapsto T, p_4 \mapsto T, p_3 \mapsto T$	$[p_3][\overline{p_3}, \overline{p_2}]$	$p_3@2$
UP	$p_1 \mapsto T, p_4 \mapsto T, p_3 \mapsto T, p_2 \mapsto F$	$[\overline{p_2}]$	$\overline{p_2}@2$

Initially no clause is unit, forcing guess to new level $p_1@1$.

Again, neither clause is unit, forcing guess to new level $p_4@2$.

Now, one clause is unit and we get a forced assignment $p_3@2$ from $p_1@1, p_4@2$ using clause $K1$.

Makes other clause unit and we get a forced assignment $\overline{p_2@2}$ from $p_3@2$ using clause $K2$.

BCP dag/implication graph: Nodes are assignments to var's (eg, $\overline{p_2}@2$). Edges labelled by antecedent clauses.

DPLL run ends in conflict

Exercise

Clause set $\langle[p_2, p_3]; [\overline{p_1}, \overline{p_4}]; [\overline{p_2}, p_4]; [\overline{p_1}, p_2, \overline{p_3}]\rangle$.

No unit, guess p_1.
Imply $\overline{p_4}$*, imply* $\overline{p_2}$*, imply* p_3*, imply* $[]$*. Conflict.*

Last guessed assignment is $p_1 \mapsto T$.
Backtrack to forced assignment $p_1 \mapsto F$.

Can guess $\overline{p_2}$ *and imply* p_3.

Example

Consider guessed assignment $\overline{x_1}, \ldots, \overline{x_{n-1}}, \overline{x_n}$, backtrack on trail and try $\overline{x_1}, \ldots, \overline{x_{n-1}}, x_n$. If the formula is unsatisfiable, every time we backtrack we will be forced to try $\overline{x_n}$.

(DLL 1962) suggested backjumping to a higher level than the previous one.

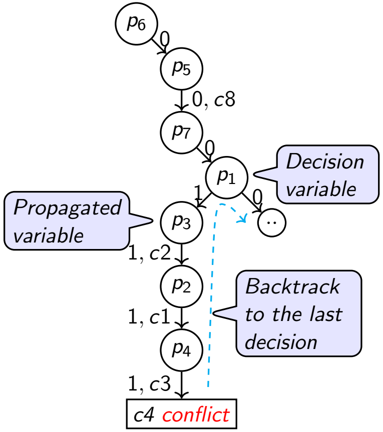
DPLL run (Ashutosh Gupta 2018)

Example: decide, propagate, and backtrack in DPLL

Example 6.4

$$
\begin{aligned}
c_1 &= (\neg p_1 \vee p_2)\\
c_2 &= (\neg p_1 \vee p_3 \vee p_5)\\
c_3 &= (\neg p_2 \vee p_4)\\
c_4 &= (\neg p_3 \vee \neg p_4)\\
c_5 &= (p_1 \vee p_5 \vee \neg p_2)\\
c_6 &= (p_2 \vee p_3)\\
c_7 &= (p_2 \vee \neg p_3 \vee p_7)\\
c_8 &= (p_6 \vee \neg p_5)
\end{aligned}
$$

Blue : causing unit propagation
Green/Blue : true clause



Exercise 6.7 *Complete the DPLL run*

Scheme (João Marques-Silva, Karem Sakallah 1996)

Represent trails with a BCP dag (implication graph):

- Nodes: partial assignments to var's (eg, $\overline{p_2}$@2)
- Edges: labelled by antecedent clauses
- At most one conflict node reached from conflicting clauses

Conflict dag: Subdag of BCP dag such that all nodes have a path to the single conflict. Non-conflict nodes have a decision literal or have literals $\overline{\ell_1}, \ldots, \overline{\ell_k}$ as predecessors where $[\ell_1, \ldots, \ell_k, \ell]$ is a clause from the given formula.

Implication graph (Ashutosh Gupta 2018)

Example: implication graph

Example 6.7

$$
\begin{aligned}
c_1 &= (\neg p_1 \vee p_2)\\
c_2 &= (\neg p_1 \vee p_3 \vee p_5)\\
c_3 &= (\neg p_2 \vee p_4)\\
c_4 &= (\neg p_3 \vee \neg p_4)\\
c_5 &= (p_1 \vee p_5 \vee \neg p_2)\\
c_6 &= (p_2 \vee p_3)\\
c_7 &= (p_2 \vee \neg p_3 \vee p_7)\\
c_8 &= (p_6 \vee \neg p_5)
\end{aligned}
$$

$$\begin{tikzcd}
p_6 \arrow[dr, "0"] & & \\
& p_5 \arrow[d, "{0, c8}"] & \\
& p_7 \arrow[dr, "0"] & \\
& & p_1 \arrow[dl, "1"'] \\
& p_3 \arrow[d, "{1, c2}"'] & \\
& p_2 \arrow[d, "{1, c1}"'] & \\
& p_4 \arrow[d, "{1, c3}"'] & \\
& \boxed{c4\ conflict} &
\end{tikzcd}$$

Implication graph

$$\begin{tikzcd}
 & \boxed{\neg p_6 @1} \arrow[d, "c8"'] & \\
\boxed{\neg p_7 @2} & \boxed{\neg p_5 @1} \arrow[d, "c2"'] & \boxed{p_1 @3} \arrow[dl, "c2"'] \arrow[d, "c1"] \\
 & \boxed{p_3 @3} \arrow[dd, "c4"'] & \boxed{p_2 @3} \arrow[d, "c3"] \\
 & & \boxed{p_4 @3} \arrow[dl, "c4"] \\
 & \boxed{conflict} &
\end{tikzcd}$$

Backjump (Richard Stallman, Gerald Sussman 1977)

Example (Where to backjump?)

$\langle K1 : [p_2, \overline{p_4}, p_5]; K2 : [\overline{p_4}, p_6, p_8]; K3 : [\overline{p_5}, \overline{p_6}, \overline{p_7}]; K4 : [\overline{p_6}, p_7]\rangle$.

Conflict dag: let conflicting clause be $p_6@5 \xrightarrow{K4} []$, $\overline{p_7}@5 \xrightarrow{K4} []$.

Let $\overline{p_7}@5$ have reasons $p_5@5 \xrightarrow{K3} \overline{p_7}@5$, $p_6@5 \xrightarrow{K3} \overline{p_7}@5$.

Let $p_6@5$ have reasons $p_4@5 \xrightarrow{K2} p_6@5$, $\overline{p_8}@3 \xrightarrow{K2} p_6@5$.

Let $p_5@5$ have reasons $p_4@5 \xrightarrow{K1} p_5@5$, $\overline{p_2}@3 \xrightarrow{K1} p_5@5$.

$K3, K4$ ($K2, K4$) do not agree on assignment of p_7 (p_6, resp'ly).
Resolve $K3, K4$ on p_7 to get $K5 : [\overline{p_5}, \overline{p_6}]$. Both $p_5@5, p_6@5$.
Resolve $K2, K5$ on p_6 to get $K6 : [\overline{p_4}, \overline{p_5}, p_8]$. Both $p_4@5, p_5@5$.
Resolve $K6, K1$ on p_5 to get $K7 : [p_2, \overline{p_4}, p_8]$. $p_4@5$ dominates paths to conflict (Edward Lowry, Clebourne Medlock 1969).

$K7$ is still conflicting. Backjumping to a level below 5, $K7$ is assertive (unit), solver flips $p_4@5$ (earlier could flip one of two variables) and learns $\overline{p_4}$. Never returns to level 5 with same partial assignment. Every resolution replaces literal by literals falsified higher up, eventually there must be an assertive literal.

Conflict-driven clause learning (Marques-Silva 1995)

$Th = \langle K1 : [p, \overline{u}]; K2 : [q, r]; K3 : [\overline{r}, w]; K4 : [u, x, y];$
$K5 : [x, \overline{y}, z]; K6 : [\overline{x}, z]; K7 : [\overline{y}, \overline{z}]; K8 : [\overline{x}, \overline{z}]; K9 : [\overline{p}, \overline{u}]\rangle$

Decision level 1: $\overline{p}@1 \xrightarrow{K1} \overline{u}@1$
Decision level 2: $\overline{q}@2 \xrightarrow{K2} r@2 \xrightarrow{K3} w@2$
Decision level 3: $\overline{x}@3 \xrightarrow{K4} y@3 \xrightarrow{K5} z@3 \xrightarrow{K7} \mathit{false}$

Resolve K5,K7 to get $K10 : [x, \overline{y}]$
Resolve K4,K10 to get $K11 : [u, x]$, which is learnt. Propagate learnt clause to backjump to:

Decision level 1: $\overline{p}@1 \xrightarrow{K1} \overline{u}@1 \xrightarrow{K11} x@1 \xrightarrow{K6} z@1 \xrightarrow{K8} \mathit{false}$

Resolve K6,K8 to get $K12 : [\overline{x}]$, which is learnt. Backjump to:

Decision level 1: $\overline{x}@1 \xrightarrow{K11} u@1 \xrightarrow{K1} p@1 \xrightarrow{K9} \mathit{false}$

Resolve K1,K9 to get $K13 : [\overline{u}]$
Resolve K13,K11 to get $K14 : [x] \xrightarrow{K12} []$

CDCL resolution proofs are not tree-like.

Certificate of unsatisfiability

A sequence of clauses $K_1 \ldots K_t$ is a certificate of unsatisfiability if $K_t = []$, and for every $1 \leq i \leq t$, the empty clause can be derived using (UP) from $Th_i = \{K_1, \ldots, K_{i-1}, \neg K_i\}$, where for $K_i = [\ell_{i,1}, \ldots, \ell_{i,n}]$, $\neg K_i \overset{\text{def}}{=} [\overline{\ell_{i,1}}] \ldots [\overline{\ell_{i,n}}]$.

Theorem (Evgenii Goldberg, Yakov Novikov 2003)

If CDCL terminates with unsatisfiability on a formula, the sequence of clauses that it has learnt is a certificate of unsatisfiability.

Sufficient to show that when clause $K = [\ell, \ell_1, \ldots, \ell_k]$ is learnt, appending the unit clauses $[\overline{\ell}][\overline{\ell_1}] \ldots [\overline{\ell_k}]$ to those already learnt will derive the empty clause by (UP).
K has been obtained from the original conflict clause by successive resolutions with reasons $\ell_{j_1}, \ldots, \ell_{j_s}$ for each literal that was removed in constructing K.

Edge-cuts (Marques-Silva 1995)

Example (Which dominator?)

$Th \supseteq \quad \langle K7 : [p_7, \overline{p_1}, p_9, p_8]; K8 : [p_7, \overline{p_4}, p_8]; K9 : [p_7, \overline{p_2}, \overline{p_3}, p_8] \rangle$

Conflict dag: let conflict clause be $p_5@6 \xrightarrow{K6} []$, $p_6@6 \xrightarrow{K6} []$.

Let $p_5@6$ have reasons $\overline{p_7}@3 \xrightarrow{K4} p_5@6$, $p_4@6 \xrightarrow{K4} p_5@6$.

Let $p_6@6$ have reasons $\overline{p_8}@3 \xrightarrow{K5} p_6@6$, $p_4@6 \xrightarrow{K5} p_6@6$.

Let $p_4@6$ have reasons $p_2@3 \xrightarrow{K3} p_4@6$, $p_3@6 \xrightarrow{K3} p_4@6$.

Let $p_2@6$ have reason $p_1@6 \xrightarrow{K1} p_2@6$.

Let $p_3@6$ have reasons $p_1@6 \xrightarrow{K2} p_2@6$, $\overline{p_9}@1 \xrightarrow{K2} p_2@6$.

Can learn $K7 : [p_7, \overline{p_1}, p_9, p_8]$ (assertive), or $K8 : [p_7, \overline{p_4}, p_8]$ (assertive), or $K9 : [p_7, \overline{p_2}, \overline{p_3}, p_8]$ (not assertive, discard).

Immediate dominator (first UIP): Choose $K8 : [p_7, \overline{p_4}, p_8]$, smaller, more likely for BCP; efficient to find (Thomas Lengauer, Robert Tarjan 1979). Different solvers make different choices.

Separation (Marques-Silva 1995)

Every edge-cut of the BCP dag separating the reason side (decision variables) of the BCP dag from the conflict side (at least one literal with its complement) is useful for learning. All nodes on the reason side having an edge going to the conflict side are the reasons of conflict. Negations of corresponding literals give the learnt clause associated with the cut.

Theorem (Marques-Silva 1995)

Learnt clauses are derived from their reasons.
Every learnt clause is a consequence of the given Th.

Proof of the first part: resolvent of consequences is a consequence. Reasons may again be learnt clauses, induction gives the second part.

Adding learnt clause to Th does not change satisfying assignments of Th.
DPLL a sound and complete algorithm, CDCL also is.

Sat solvers in practice

For large *Th*, huge number of clauses can be learnt in an execution. How to manage this database of lemmas?

- CDCL solvers perform well: lakh variables, million clauses
- Aggressively erase learnt clauses that do not seem useful
- Sometimes they keep learnt clauses but restart search
- Tricky formulas like propositional Pigeonhole Principle (Haken 1985) and propositional Ordering Principle (Balakrishnan Krishnamurthy 1985) may not appear
- Several parameters studied by theoreticians may only coarsely reflect divisions found in practice. For example clause-variable ratio: number of clauses in formula divided by number of variables in formula, for random formulas at about 4.26 there is a “phase transition” from almost-certainly-satisfiable to almost-certainly-unsatisfiable
- Heuristics are far ahead of theory

Parameters (Vijay Ganesh, Moshe Vardi 2020)

(Resolution) on pivot A: $\langle[B, A]; [\neg A, C]\rangle \implies [B, C]$ resolvent.

Merge: number of overlapping literals in B, C.
Heuristic (Peter Andrews 1968): Choose clauses which have a high amount of merge.
Studied by (Edward Zulkoski, Ruben Martins, Christoph Wintersteiger, Jia Hui Liang, Krzysztof Czarnecki, Vijay Ganesh 2018).

Modularity: incidence graph of a formula (nodes variables, edge if appearing in same clause).
Heuristic (Carlos Ansótegui, Jesús Giráldez-Cru, Jordi Levy 2012): If incidence graph has “clusters” weakly connected to other clusters, solver can work on “decomposition” of formula.
Studied by (Zack Newsham, Vijay Ganesh, Sebastian Fischmeister, Gilles Audemard, Laurent Simon 2014).