Unit Propagate:

$M \parallel N \cup \{C \vee L\} \;\Rightarrow_{\text{DPLL}}\; M\,L \parallel N \cup \{C \vee L\}$

if C is false under M and L is undefined under M.

Decide:

$M \parallel N \;\Rightarrow_{\text{DPLL}}\; M\,L^{\text{d}} \parallel N$

if L is undefined under M and contained in N.

Fail:

$M \parallel N \cup \{C\} \;\Rightarrow_{\text{DPLL}}\; \mathit{fail}$

if C is false under M and M contains no decision literals.

Backjump:

$M'\,L^{\text{d}}\,M'' \parallel N \;\Rightarrow_{\text{DPLL}}\; M'\,L' \parallel N$

if there is some "backjump clause" $C \vee L'$ such that
$N \models C \vee L'$,
C is false under M', and
L' is undefined under M'.

We will see later that the Backjump rule is always applicable, if the list of literals M contains at least one decision literal and some clause in N is false under M.

There are many possible backjump clauses. One candidate: $\overline{L_1} \vee \ldots \vee \overline{L_n}$, where the L_i are all the decision literals in $M\,L^{\text{d}}\,M'$. (But usually there are better choices.)

Lemma 1.14 *If we reach a state $M \parallel N$ starting from $\emptyset \parallel N$, then:*

(1) M does not contain complementary literals.

(2) Every deduced literal L in M follows from N and decision literals occurring before L in M.

Proof. By induction on the length of the derivation. □

Lemma 1.15 *Every derivation starting from $\emptyset \parallel N$ terminates. (Proof follows)*

Proof. (Idea) Consider a DPLL derivation step $M \parallel N \;\Rightarrow_{\text{DPLL}}\; M' \parallel N'$ and a decomposition $M_0 l_1^d M_1 \ldots l_k^d M_k$ of M (accordingly for M'). Let n be the number of distinct propositional variables in N. Then k, k' and the length of M, M' are always smaller than n. We define $f(M) = n - \text{length}(M)$ and finally

$$M \parallel N \;\succ\; M' \parallel N' \quad \text{if}$$

(i) $f(M_0) = f(M'_0), \ldots, f(M_{i-1}) = f(M'_{i-1}), f(M_i) > f(M'_i)$ for some $i < k, k'$ or

(ii) $f(M_j) = f(M'_j)$ for all $1 \leq j \leq k$ and $f(M) > f(M')$.

Lemma 1.16 *Suppose that we reach a state $M \parallel N$ starting from $\emptyset \parallel N$ such that some clause $D \in N$ is false under M. Then:*

(1) If M does not contain any decision literal, then "Fail" is applicable.

(2) Otherwise, "Backjump" is applicable.

(Proof follows)

Proof. (1) Obvious.

(2) Let $L_1, \ldots, L_n$ be the decision literals occurring in M (in this order). Since $M \models \neg D$, we obtain, by Lemma 1.14, $N \cup \{L_1, \ldots, L_n\} \models \neg D$. Since $D \in N$, $N \models \overline{L_1} \vee \cdots \vee \overline{L_n}$. Now let $C = \overline{L_1} \vee \cdots \vee \overline{L_{n-1}}$, $L' = \overline{L_n}$, $L = L_n$, and let M' be the list of all literals of M occurring before L_n, then the condition of "Backjump" is satisfied. □

Theorem 1.17 *(1) If we reach a final state $M \parallel N$ starting from $\emptyset \parallel N$, then N is satisfiable and M is a model of N.*

(2) If we reach a final state fail starting from $\emptyset \parallel N$, then N is unsatisfiable.

(Proof follows)

Proof. (1) Observe that the "Decide" rule is applicable as long as literals are undefined under M. Hence, in a final state, all literals must be defined. Furthermore, in a final state, no clause in N can be false under M, otherwise "Fail" or "Backjump" would be applicable. Hence M is a model of every clause in N.

(2) If we reach *fail*, then in the previous step we must have reached a state $M \parallel N$ such that some $C \in N$ is false under M and M contains no decision literals. By part (2) of Lemma 1.14, every literal in M follows from N. On the other hand, $C \in N$, so N must be unsatisfiable. □

Getting Better Backjump Clauses

Suppose that we have reached a state $M \parallel N$ such that some clause $C \in N$ (or following from N) is false under M.

Consequently, every literal of C is the complement of some literal in M.

(1) If every literal in C is the complement of a decision literal of M. Then C is a backjump clause.

(2) Otherwise, $C = C' \vee \overline{L}$, such that L is a deduced literal.

For every deduced literal L, there is a clause $D \vee L$, such that $N \models D \vee L$ and D is false under M.

Consequently, $N \models D \vee C'$ and $D \vee C'$ is also false under M.

By repeating this process, we will eventually obtain a clause that consists only of complements of decision literals and can be used in the "Backjump" rule.

Moreover, such a clause is a good candidate for learning.

Learning Clauses

The DPLL system can be extended by two rules to learn and to forget clauses:

Learn:

$M \parallel N \;\Rightarrow_{\mathrm{DPLL}}\; M \parallel N \cup \{C\}$

if $N \models C$.

Forget:

$M \parallel N \cup \{C\} \;\Rightarrow_{\mathrm{DPLL}}\; M \parallel N$

if $N \models C$.

If we ensure that no clause is learned infinitely often, then termination is guaranteed.

The other properties of the basic DPLL system hold also for the extended system.

Preprocessing

Modern SAT solvers use the following techniques:

(i) Subsumption
(ii) Purity Deletion
(iii) Merging Replacement Resolution
(iv) Tautology Deletion
(v) Literal Elimination: do all possible resolution step on a literal and throw away the parent clauses

Further Information

The ideas described so far heve been implemented in all modern SAT solvers: *zChaff*, *miniSAT*,*picoSAT*. Because of clause learning the algorithm is now called CDCL: Conflict Driven Clause Learning.

It has been shown in 2009 that CDCL can polynomially simulate resolution, a long standing open question:

Knot Pipatsrisawat, Adnan Darwiche : On the Power of Clause-Learning SAT Solvers with Restarts. CP 2009 : 654-668

Literature:

Lintao Zhang and Sharad Malik: The Quest for Efficient Boolean Satisfiability Solvers; Proc. CADE-18, LNAI 2392, pp. 295–312, Springer, 2002.

Robert Nieuwenhuis, Albert Oliveras, Cesare Tinelli: Solvin SAT and SAT Modulo Theories; From an abstract Davis-Putnam-Logemann-Loveland precedure to DPLL(T), pp 937–977, Journal of the ACM, 53(6), 2006.

Armin Biere, Marijn Heule, Hans van Maaren, Toby Walsh (Editors): Handbook of Satisfiability; IOS Press, 2009

Daniel Leberre's slides at VTSA'09: `http://www.mpi-inf.mpg.de/vtsa09/`.