Clause representation of CNF formulas

- Clause: set of literals (disjunction).

$$\{A, B\} \text{ stands for } (A \vee B).$$

- Formula: set of clauses (conjunction).

$$\{\{A, B\}, \{\neg A, B\}\} \text{ stands for } ((A \vee B) \wedge (\neg A \vee B)).$$

The empty clause stands for **false**.
The empty formula stands for **true**.

The DPLL algorithm

- Developed by Davis, Putman, Loveland und Logemann
- Basis for the most efficient of today's solvers.
- Based on the following ideas for checking satisfiability of a CNF formula F:
  - If F is empty, then F is satisfiable.
  - If F contains the empty clause, then F is not satisfiable.
  - If $\{L\} \in F$ for some literal L, then whenever $\mathcal{A} \models F$ we must have $\mathcal{A} \models L$ (and we can remove every clause containing L and every occurrence of $\overline{L}$ in any clause of F).
  - If a literal L appears nowhere in F, then we can safely remove each clause of F that contains $\overline{L}$.
  - Otherwise we apply brute-force.

Removing literals and suitable clauses

Let L be a literal that appears in a CNF formula F.

By F_L we denote the formula one obtains from F by

- removing each clause of F that contains L and
- removing each occurrence of $\overline{L}$ in F.

Lemma (exercise) Let L be a literal that appears in F. Then the following holds:

(L1) If $\{L\} \in F$, then F is satisfiable if and only if F_L is satisfiable.

(L2) If L appears in F but not $\overline{L}$, then F is satisfiable if and only if F_L is satisfiable.

(L3) F is satisfiable if and only if F_L is satisfiable or $F_{\overline{L}}$ is satisfiable.

Davis-Putnam-Logemann-Loveland algorithm

procedure DPLL(F)

    if F is the empty formula **then return** "yes"

    else if F contains the empty clause **then return** "no"

    else if $\{L\} \in F$ for some literal L **then return** DPLL(F_L)

    else if L appears in F but not $\overline{L}$ **then return** DPLL(F_L)

    else select literal L appearing in F

        if DPLL(F_L)$=$ "yes" **then return** "yes"

        else return DPLL($F_{\overline{L}}$)

The DPLL algorithm: Termination and complexity

The weight of a clause $C = \{L_1, \ldots, L_k\}$ is defined as $\|C\| = k$.

The weight of a CNF formula $F = \{C_1, \ldots, C_k\}$ is defined as $\|F\| = k + \sum_{i=1}^{k} \|C_i\|$.

Note that $\|F\| \leq 2|F|$.

Lemma: If on input F the DPLL algorithm recursively calls DPLL(G), then $\|G\| < \|F\|$.

Proof: Simple inspection of the DPLL algorithm and definition of F_L for literals L.

Corollary: Since $\|F\| = 0$ iff F is the empty formula, it follows that the recursion depth of the DPLL algorithm on input F is bounded by $2|F|$.

Corollary: The running time of the DPLL algorithm on input F is bounded by $2^{O(|F|)}$.

The DPLL algorithm: Correctness

Lemma: On input F the DPLL algorithm returns "yes" if and only if F is satisfiable.

Proof: By induction on the recursion depth of algorithm DPLL on input F.

Induction base. If the recursion depth is 0, then either (i) F is the empty formula and thus satisfiable; DPLL returns "yes" in this case, as required, or (ii) F contains an empty clause and thus F is unsatisfiable; DPLL returns "no" in this case, as required.

Induction step. The cases when $\{L\} \in F$ for some literal L or when L appears in F but not $\overline{L}$ for some literal L follow from induction hypothesis and (L2) and (L3), respectively.

The remaining case (F is satisfiable if and only F_L is satisfiable or $F_{\overline{L}}$ is satisfiable) follows from induction hypothesis and (L3).