Example

Continuing the example from the previous lecture, we first transform the given formulas into logically equivalent disjunctions of literals (atomic formulas or negations thereof):

$$\begin{aligned}
\neg A \lor \neg B \lor \neg C \lor D \\
A \lor \neg M \lor L \\
\neg F \lor \neg E \lor \neg D \\
\neg G \lor \neg M \lor C \\
B \lor \neg F \lor \neg H \\
D \lor \neg B \lor \neg E \lor G \\
\neg M \lor I \lor J \\
\neg H \lor \neg M \lor K \\
\neg K \lor \neg J \lor L \lor E \\
H \lor \neg F \lor L \\
\neg M \lor \neg L \lor \neg F \\
\neg K \lor \neg I \lor \neg A \lor E
\end{aligned}$$

We want to determine whether $M \to \neg F$ is a logical consequence of these formulas.

This problem is equivalent to determining whether the formula $S \to (M \to \neg F)$ is a tautology, where S is the conjunction of the above disjunctions.

This tautology problem in turn is equivalent to determining whether the negated formula, $\neg(S \to (M \to \neg F))$, is a contradiction.

The latter formula is equivalent to $S \wedge M \wedge F$. We will attempt to show that this formula is unsatisfiable, i.e., a contradiction.

Example (continued)

Let S^0 denote the formula $S \wedge M \wedge F$.

We attempt to show unsatisfiability by arguing that there can be no truth valuation that satisfies S^0.

First note that if a valuation satisfies S^0 it must assign $\top$ to both M and F.

We may use this observation to simplify subformulas of S^0 as follows:

$$\begin{array}{r}
\neg A \vee \neg B \vee \neg C \vee D \\
A \vee L \\
\neg E \vee \neg D \\
\neg G \vee C \\
B \vee \neg H \\
D \vee \neg B \vee \neg E \vee G \\
I \vee J \\
\neg H \vee K \\
\neg K \vee \neg J \vee L \vee E \\
H \vee L \\
\neg L \\
\neg K \vee \neg I \vee \neg A \vee E
\end{array}$$

Let S^1 be the conjunction of the above formulas and of the two literals M and F.

It can easily be seen that S^1 is satisfiable if, and only if, S^0 is satisfiable.

Example (continued)

Next observe that S^1 is only satisfiable under a valuation in which $\neg L$ is true or, equivalently, L is false.

This allows further simplifications as we may delete those disjunctions containing $\neg L$ (which are true when L is false) and drop L from all other disjunctions:

$$\begin{array}{r}
\neg A \vee \neg B \vee \neg C \vee D \\
A \\
\neg E \vee \neg D \\
\neg G \vee C \\
B \vee \neg H \\
D \vee \neg B \vee \neg E \vee G \\
I \vee J \\
\neg H \vee K \\
\neg K \vee \neg J \vee E \\
H \\
\neg K \vee \neg I \vee \neg A \vee E
\end{array}$$

Let S^2 be the conjunction of the above formulas and the literals M, F, and $\neg L$.

The formula S^2 is satisfiable if, and only if, S^1 is satisfiable.

Example (continued)

What next?

Eliminate A and H:

$$\begin{aligned} \neg B \vee \neg C \vee D \\ \neg E \vee \neg D \\ \neg G \vee C \\ B \\ D \vee \neg B \vee \neg E \vee G \\ I \vee J \\ K \\ \neg K \vee \neg J \vee E \\ \neg K \vee \neg I \vee E \end{aligned}$$

Continue by getting rid of B and K:

$$\begin{aligned} \neg C \vee D \\ \neg E \vee \neg D \\ \neg G \vee C \\ D \vee \neg E \vee G \\ I \vee J \\ \neg J \vee E \\ \neg I \vee E \end{aligned}$$

At this point we have a conjunction of the above seven formulas and the literals M, F, $\neg L$, A, H, B, and K.

Example (continued)

We pick one variable, say D, and consider two cases.

(i) If $D = \top$ we can simplify as follows:

$$\begin{aligned} \neg E \\ \neg G \vee C \\ I \vee J \\ \neg J \vee E \\ \neg I \vee E \end{aligned}$$

Then we may eliminate E, to get:

$$\begin{aligned} \neg G \vee C \\ I \vee J \\ \neg J \\ \neg I \end{aligned}$$

At this point we can get rid of I or J, but then obtain a contradiction (the formula $\perp$). Let us therefore consider the alternative case.

(ii) If $D = \perp$, the simplification results in:

$$\begin{aligned} \neg C \\ \neg G \vee C \\ \neg E \vee G \\ I \vee J \\ \neg J \vee E \\ \neg I \vee E \end{aligned}$$

We can eliminate C:

$$\begin{aligned} \neg G \\ \neg E \lor G \\ I \lor J \\ \neg J \lor E \\ \neg I \lor E \end{aligned}$$

and then G:

$$\begin{aligned} \neg E \\ I \lor J \\ \neg J \lor E \\ \neg I \lor E \end{aligned}$$

and then E:

$$\begin{aligned} I \lor J \\ \neg J \\ \neg I \end{aligned}$$

and finally I or J to obtain a contradiction again.

What is the conclusion?

In each case we have shown that S^0 is (equivalent to) a contradiction. Consequently, $M \to \neg F$ is a logical consequence of S.

Proof Methods for Propositional Logic

The validity of a propositional reasoning argument can in principle be checked by examining a suitable truth table. But this approach is impractical if the number of propositional variables that need to be considered is large.

The method implicit in the above example is called the *Davis-Putnam method*.

It is based on testing whether the set of all premises plus the negation of the conclusion is unsatisfiable; and in fact is one of the fastest methods in practice for solving the (propositional) satisfiability problem.

The Davis-Putnam Method

A literal is a propositional variable P (a *positive* literal) or the negation thereof, $\neg P$ (a *negative* literal).

The *complement* of a positive literal P is the literal $\neg P$; the complement of $\neg P$ is P. A (generalized) disjunction of literals is also called a *clause*.

The Davis-Putnam method accepts as input a finite set of clauses N, and determines whether (the conjunction of all formulas of) N is satisfiable or not.

Formally, the method can be described as a process of constructing a binary tree, the nodes of which are labelled by sets of clauses.

The initial tree consists of a single node labelled by N. A tree can be expanded by adding children to a selected leaf according to specified "tree expansion rules" of the form $S \vdash S'$ or $S \vdash S'_1|S'_2$.

If a leaf in a tree is labelled by a set of clauses that matches the description of S, one adds either one child labelled by S', or else two children labelled by S'_1 and S'_2, respectively, depending on which kind of rule is applied.

Repeated application of rules thus induces a sequence of ever larger binary trees.

Tree Expansion Rules

If N is a set of clauses and C a clause, we denote by N, C the set of clauses $N \cup \{C\}$.

Unit reduction [R]

$$N, L, C \vee L' \vdash N, L, C$$

where L' is the complement of L and N does not contain $C \vee L'$.

Unit subsumption [S]

$$N, L, C \vee L \vdash N, L$$

where N does not contain $C \vee L$.

Tautology Deletion [D]

$$N, C \vdash N$$

where C is a tautology and N does not contain C.

Elimination [E]

$$N, L, L' \vdash \{\bot\}$$

where L' is the complement of L

Splitting

The following key rule expresses a form of "case analysis."

Splitting [P]

$$N \vdash N, P \mid N, \neg P$$

where the variable P occurs in a non-unit clause in N, but not in a unit clause.

This rule is the only one that produces two children; all other rules generate a single child.

The following (final) rule is optional:

Pure literal extension [L]

$$N \vdash N, L$$

where (i) the clause L itself is not an element of N and (ii) the literal L occurs in one of the clauses in N, but its complement does not occur as a literal of a clause in N.

Soundness

The soundness of the Davis-Putnam method is based on the observation that all expansion rules preserve satisfiability in a certain sense.

Lemma

> If a node labelled by N' is expanded by a non-splitting rule to a node labelled by N'', then N' is satisfiable if, and only if, N'' is satisfiable.
>
> If a node labelled by N' is expanded by the splitting rule to nodes labelled by N''_1 and N''_2, respectively, then N' is satisfiable if, and only if, N''_1 or N''_2 is satisfiable.

The lemma can be proved by examining the different rules.

The following theorem can be proved by induction on the size of a tree, and using the above lemma.

Theorem

> Let N be the label of the root of a tree that has been constructed by applying Davis-Putnam expansion rules. Then N is satisfiable if, and only if, at least one leaf is labelled by a satisfiable set.

Expansion Strategies

The tree expansion rules define the basic steps of the Davis-Putnam method. To completely define a procedure for testing satisfiability of a set of clauses, one needs to specify which rule is to be applied at any given point, and where (i.e., to which node).

The collection of rules in essence specifies a *class of procedures* that are based on common elementary computation steps, but differ in their underlying computation strategies.

For instance, one sensible strategy is to apply unit reduction and subsumption, as well as tautology deletion and elimination exhaustively, but apply splitting only when necessary (i.e., when no other rule can be applied).

Pure literal extension may be considered as an optional rule, to be used in connection with unit subsumption, a combination also known as *pure literal elimination*.

A key property of the above set of expansion rules is that each strategy is guaranteed to terminate.

Termination

A *unit* clause is a clause with just one literal. Thus, P and $\neg Q$ are unit clauses, whereas $P \vee \neg Q$ is not.

Let T be a tree constructed via Davis-Putnam rules from an initial, finite set of clauses N and let k be the number of different propositional variables occurring in N.

We assign to each node in T a *weight* that depends on the label S of that node and is defined as follows:

$$w(S) = \begin{cases} (0,0) & \text{if } S = \{\bot\} \\ (k - n_S, l_S) & \text{otherwise} \end{cases}$$

where n_S is the number of different variables which occur in a unit clause in S, and l_S is the size (or length) of S.

Pairs of numbers can be compared lexicographically:

$$(i, j) \succ (i', j')$$

if, and only if, either $i > i'$ or else $i = i'$ and $j > j'$.

Lemma

> Let T be a tree constructed according to the Davis-Putnam rules. If S' is the (label of the) child of (a node labelled by) S, then $w_S \succ w_{S'}$.

The Infinity Lemma

The above lemma implies that a tree constructed according to Davis-Putnam rules has no infinite branches, as the weights of nodes are decreasing along each branch and there can be no infinitely decreasing sequence of weights.

These trees are also finitely branching and we may therefore apply the following result.

König's Infinity Lemma

> A finitely branching, infinite tree has an infinite branch.

We claim that there is no infinite sequence of trees,

$$T_0, T_1, T_2, \ldots$$

such that each tree T_{i+1} is obtained from T_i by application of a Davis-Putnam rule.

For if there were such a sequence, it would define a finitely branching, infinite tree $\bigcup_i T_i$, which has no infinite branches. This would contradict the Infinity Lemma.

We may conclude that the tree expansion process always terminates.

A Decision Procedure for Satisfiability

If no further expansion rules can be applied to a node labelled by a set of clauses N, then either

1. N is $\{\bot\}$ (and hence is unsatisfiable) or

2. N contains only unit clauses (and is satisfiable).

In the second case one can extract a truth assignment σ from N as follows:

$$P\sigma = \begin{cases} \top & \text{if } N \text{ contains unit clause } P \\ \bot & \text{otherwise} \end{cases}$$

Note that N contains no two clauses P and $\neg P$, for otherwise the elimination rule can be applied. Consequently all clauses in N are true under σ, and hence the set labelling the root of the tree is also satisfiable.

If in a tree (to which no further expansion rules are applicable) all leaves are labelled by $\{\bot\}$, then the set of clauses labelling the root of that tree is unsatisfiable.

In sum, the Davis-Putnam method yields a *decision procedure* for the problem of determining whether a (finite) conjunction of propositional clauses is satisfiable or not.

Horn Clauses

A clause is a (generalized) disjunction of literals (positive and/or negative). If a clause contains at most one positive literal it is called a *Horn clause*.

In other words, Horn clauses contain either one or no positive literals. Thus, they are of the form

$$P$$

(a single positive literal), or

$$\neg P_1 \vee \cdots \vee \neg P_n \vee Q$$

(one positive literal and some negative literals), or

$$\neg P_1 \vee \cdots \vee \neg P_n$$

(only negative literals).

Such Horn clauses are often written as implications, e.g,

$$P_1 \wedge \cdots \wedge P_n \rightarrow Q$$

or

$$P_1 \wedge \cdots \wedge P_n \rightarrow \bot,$$

in which case the negation symbol will not occur explicitly.

Satisfiability of Horn Clauses

Satisfiability of conjunctions of Horn clauses can be determined efficiently, for instance, by a variant of the Davis-Putnam method.

Let H be a set of Horn clauses. Construct a tree for H by applying the following expansion rules:

- tautology deletion,
- elimination, and
- a restricted version of unit reduction,

  $$N, P, C \vee \neg P \vdash N, P, C$$

  where P is an atomic formula (i.e., a propositional variable).

This last rule is also called *positive unit reduction*. No other rules are applied.

An exhaustive application of the above rules necessarily terminates and, since the splitting rule is never used, will result in a "linear" tree with a single branch.

By inspecting the leaf of this tree one can determine whether the initial set H is satisfiable or not.

The Original Davis-Putnam Method

The method we have described, though often called the "Davis-Putnam" procedure in the research literature, is actually a variation, published by Davis, Logemann and Loveland in 1962, of the original method proposed by Davis and Putnam in 1960.

The variation is more efficient than the original method for practical purposes.

The original Davis-Putnam method can be described in terms of the following rule:

Davis-Putnam self-resolution

$$\frac{F}{F[P/\bot] \vee F[P/\top]}$$

where F is a formula in conjunctive form, P an atom occurring in F, and $F[P/\bot]$ and $F[P/\top]$ denote the formulas obtained from F by replacing all occurrence of P by $\bot$ and $\top$, respectively.

This rule expresses a "case split" via disjunction.

This method operates on a single (possibly large) formula F, which corresponds to the conjunction of all clauses in the clause sets we used above.

Note that the premise F of the above rule is assumed to be in conjunctive form, whereas the conclusion is not. Thus, suitable simplification rules for transforming the conclusion to conjunctive form have to be applied before the above self-resolution rule can be applied again.

Rules such as unit reduction and unit subsumption can be easily adapted to this context.

Simplification of Formulas

Proof methods, such as the Davis-Putnam method, may require input formulas to be in a certain format. Processing of formulas that are not in the right format can often be done by simplification via equivalences.

More specifically, we may use equivalences to repeatedly replace subformulas by equivalent, yet simpler, formulas until the desired form is obtained.

For instance, certain connectives can be eliminated. The equivalence,

$$P \rightarrow Q \sim \neg P \vee Q,$$

when used from left to right, produces simpler formulas, provided one prefers negation and disjunction to implication.

The following equivalences,

$$\begin{array}{rcl} \neg\top & \sim & \bot \\ \neg\bot & \sim & \top \\ P \wedge \top & \sim & P \\ \top \wedge P & \sim & P \\ P \wedge \bot & \sim & \bot \\ \bot \wedge P & \sim & \bot \\ P \vee \top & \sim & \top \\ \top \vee P & \sim & \top \\ P \vee \bot & \sim & P \\ \bot \vee P & \sim & P \end{array}$$

can be used to eliminate all occurrences of $\top$ and $\bot$ as proper subformulas.

Negation Normal Form

A key requirement for the simplification process is that it *terminate*.

Termination may be obvious, as in the above examples, or may require more subtle reasoning.

Consider for example the following rules:

$$\begin{array}{rcl} \neg\neg P & \sim & P \\ \neg(P \wedge Q) & \sim & \neg P \vee \neg Q \\ \neg(P \vee Q) & \sim & \neg P \wedge \neg Q \end{array}$$

Does their repeated application (of replacing an instance of a left-hand side by the corresponding instance of the right-hand side) necessarily terminate?

Yes. Informally, negation symbols are pushed to the inside, a process that has to terminate. This can be formally proved by assigning a suitable "weight" to formulas that decreases with each replacement step.

The above rules can be used to generate a *negation normal form* of a propositional formula.

Conjunctive and Disjunctive Forms

Let us add to the simplification rules for negation normal form the following distributivity rules:

$$\begin{array}{rcl} P \vee (Q \wedge R) & \sim & (P \vee Q) \wedge (P \vee R) \\ (P \wedge Q) \vee R & \sim & (P \vee R) \wedge (Q \vee R) \end{array}$$

Exhaustive application of the extended set of rules results in formulas in *conjunctive form*.

Disjunctive forms can be obtained similar way, but using the equivalences for distributing conjunction over disjunction instead.

Conjunctive forms are usually written as generalized conjunctions,

$$C_1 \wedge \cdots \wedge C_n,$$

where each conjunct C_i is a (generalized) disjunction

$$L_{i1} \vee \cdots \vee L_{ik_i},$$

and each formula $L_{i,j}$ is a *literal*, that is, a proposition variable P or the negation thereof, $\neg P$.

Sometimes one speaks of a *conjunctive normal form in variables* $P_1, \ldots, P_n$ if each formula C_i is a *constituent*, i.e., contains either P_i or $\neg P_i$, for all i with $1 \leq i \leq n$.

For example, the formula $(P \vee Q) \wedge \neg P$ is in conjunctive form, but not a normal form in this sense, because the second basic conjunction is not a constituent (it contains only $\neg P$).

An equivalent normal form is

$$(P \vee Q) \wedge (\neg P \vee Q) \wedge (\neg P \vee \neg Q).$$

Disjunctive normal forms are defined in a similar way.