COMP30026 Models of Computation

Symbolic Deduction

Harald Søndergaard

Lecture Week 3 Part 1 (Zoom)

Semester 2, 2020

This Lecture is Being Recorded

Mechanising Deduction

> *We must not think that computation . . . has place only in numbers.*
>
> — *T. Hobbes (1655)*

Calculus ratiocinator: G. W. Leibniz (1679).

Propositional logic:
G. Boole, A. De Morgan, E. Schröder (19th century).

Predicate logic: G. Frege (1879).

Universal computers:
C. Babbage (19th century), A. Turing (20th century).

Propositional Logic is Decidable

Generating a truth table is just a way of systematically exploring the truth value of a formula, for each possible truth assignment, that is, using brute force.

Formulas are assumed to be finite, so this way we can always decide if a formula is satisfiable, and whether it is valid.

Unfortunately truth tables may grow exponentially in the size of formulas.

Faster Satisfiability/Validity Checking?

Are there faster decision procedures for propositional logic?

It depends on how we choose to write the formulas.

What if we don't want to commit to any particular form?

Then satisfiability is NP-complete and validity is co-NP-complete.

These are terms from complexity theory. Historically the satisfiability problem for propositional logic, SAT, has been seminal for this theory.

SAT and Complexity Theory

A who's who of important problems in scheduling, network flow and routing, database management, number theory, automata, circuit fault detection, code generation, ..., have been shown to be tractable if and only if SAT is.

Most computer scientists conclude from this that it is unlikely that there are decision procedures for SAT (and hence for all those other problems) that perform much better than brute force.

Normal Forms for Propositional Logic

A literal is P or $\neg P$ where P is a propositional letter.

A formula is in conjunctive normal form (CNF) if it is a conjunction of disjunctions of literals (a conjunction of "clauses").

$$(A \vee \neg B) \wedge (B \vee C \vee D) \wedge A$$

It is in disjunctive normal form (DNF) if it is a disjunction of conjunctions of literals.

$$(\neg A \wedge \neg B) \vee (\neg B \wedge C) \vee (A \wedge \neg D)$$

Theorem: Every propositional formula can be expressed in CNF, as well as in DNF.

Converting a Formula to CNF or to DNF

1. Eliminate all occurrences of $\oplus$, using $A \oplus B \equiv (A \vee B) \wedge (\neg A \vee \neg B)$.
2. Eliminate all occurrences of $\Leftrightarrow$, using $A \Leftrightarrow B \equiv (A \Rightarrow B) \wedge (B \Rightarrow A)$.
3. Eliminate all occurrences of $\Rightarrow$ using $A \Rightarrow B \equiv \neg A \vee B$.
4. Use De Morgan's Laws to push $\neg$ inward over $\wedge$ and $\vee$.
5. Eliminate double negations using $\neg\neg A \equiv A$.
6. Use the distributive laws to get the required form.

Example Conversion to CNF

$$(\neg P \wedge (\neg Q \Rightarrow R)) \Leftrightarrow S$$

Example Conversion to CNF

$$
\begin{aligned}
& (\neg P \wedge (\neg Q \Rightarrow R)) \Leftrightarrow S \\
\equiv \quad & ((\neg P \wedge (\neg Q \Rightarrow R)) \Rightarrow S) \wedge (S \Rightarrow (\neg P \wedge (\neg Q \Rightarrow R))) \qquad (2)
\end{aligned}
$$

Example Conversion to CNF

$$
\begin{aligned}
& (\neg P \wedge (\neg Q \Rightarrow R)) \Leftrightarrow S \\
\equiv \quad & ((\neg P \wedge (\neg Q \Rightarrow R)) \Rightarrow S) \wedge (S \Rightarrow (\neg P \wedge (\neg Q \Rightarrow R))) && (2) \\
\equiv \quad & (\neg(\neg P \wedge (\neg Q \Rightarrow R)) \vee S) \wedge (\neg S \vee (\neg P \wedge (\neg Q \Rightarrow R))) && (3)
\end{aligned}
$$

Example Conversion to CNF

$$
\begin{aligned}
& (\neg P \wedge (\neg Q \Rightarrow R)) \Leftrightarrow S \\
\equiv\ & ((\neg P \wedge (\neg Q \Rightarrow R)) \Rightarrow S) \wedge (S \Rightarrow (\neg P \wedge (\neg Q \Rightarrow R))) && (2) \\
\equiv\ & (\neg(\neg P \wedge (\neg Q \Rightarrow R)) \vee S) \wedge (\neg S \vee (\neg P \wedge (\neg Q \Rightarrow R))) && (3) \\
\equiv\ & (\neg(\neg P \wedge (\neg\neg Q \vee R)) \vee S) \wedge (\neg S \vee (\neg P \wedge (\neg\neg Q \vee R))) && (3)
\end{aligned}
$$

Example Conversion to CNF

$$
\begin{aligned}
& (\neg P \wedge (\neg Q \Rightarrow R)) \Leftrightarrow S \\
\equiv\ & ((\neg P \wedge (\neg Q \Rightarrow R)) \Rightarrow S) \wedge (S \Rightarrow (\neg P \wedge (\neg Q \Rightarrow R))) && (2) \\
\equiv\ & (\neg(\neg P \wedge (\neg Q \Rightarrow R)) \vee S) \wedge (\neg S \vee (\neg P \wedge (\neg Q \Rightarrow R))) && (3) \\
\equiv\ & (\neg(\neg P \wedge (\neg\neg Q \vee R)) \vee S) \wedge (\neg S \vee (\neg P \wedge (\neg\neg Q \vee R))) && (3) \\
\equiv\ & ((\neg\neg P \vee (\neg\neg\neg Q \wedge \neg R)) \vee S) \wedge (\neg S \vee (\neg P \wedge (\neg\neg Q \vee R))) && (4)
\end{aligned}
$$

Example Conversion to CNF

$$
\begin{aligned}
& (\neg P \wedge (\neg Q \Rightarrow R)) \Leftrightarrow S \\
\equiv\ & ((\neg P \wedge (\neg Q \Rightarrow R)) \Rightarrow S) \wedge (S \Rightarrow (\neg P \wedge (\neg Q \Rightarrow R))) && (2) \\
\equiv\ & (\neg(\neg P \wedge (\neg Q \Rightarrow R)) \vee S) \wedge (\neg S \vee (\neg P \wedge (\neg Q \Rightarrow R))) && (3) \\
\equiv\ & (\neg(\neg P \wedge (\neg\neg Q \vee R)) \vee S) \wedge (\neg S \vee (\neg P \wedge (\neg\neg Q \vee R))) && (3) \\
\equiv\ & ((\neg\neg P \vee (\neg\neg\neg Q \wedge \neg R)) \vee S) \wedge (\neg S \vee (\neg P \wedge (\neg\neg Q \vee R))) && (4) \\
\equiv\ & ((P \vee (\neg Q \wedge \neg R)) \vee S) \wedge (\neg S \vee (\neg P \wedge (Q \vee R))) && (5)
\end{aligned}
$$

Example Conversion to CNF

$$
\begin{aligned}
& (\neg P \wedge (\neg Q \Rightarrow R)) \Leftrightarrow S \\
\equiv\ & ((\neg P \wedge (\neg Q \Rightarrow R)) \Rightarrow S) \wedge (S \Rightarrow (\neg P \wedge (\neg Q \Rightarrow R))) && (2) \\
\equiv\ & (\neg(\neg P \wedge (\neg Q \Rightarrow R)) \vee S) \wedge (\neg S \vee (\neg P \wedge (\neg Q \Rightarrow R))) && (3) \\
\equiv\ & (\neg(\neg P \wedge (\neg\neg Q \vee R)) \vee S) \wedge (\neg S \vee (\neg P \wedge (\neg\neg Q \vee R))) && (3) \\
\equiv\ & ((\neg\neg P \vee (\neg\neg\neg Q \wedge \neg R)) \vee S) \wedge (\neg S \vee (\neg P \wedge (\neg\neg Q \vee R))) && (4) \\
\equiv\ & ((P \vee (\neg Q \wedge \neg R)) \vee S) \wedge (\neg S \vee (\neg P \wedge (Q \vee R))) && (5) \\
\equiv\ & (((P \vee \neg Q) \wedge (P \vee \neg R)) \vee S) \\
& \quad \wedge ((\neg S \vee \neg P) \wedge (\neg S \vee (Q \vee R))) && (6)
\end{aligned}
$$

Example Conversion to CNF

$$
\begin{aligned}
& (\neg P \wedge (\neg Q \Rightarrow R)) \Leftrightarrow S \\
\equiv\ & ((\neg P \wedge (\neg Q \Rightarrow R)) \Rightarrow S) \wedge (S \Rightarrow (\neg P \wedge (\neg Q \Rightarrow R))) && (2) \\
\equiv\ & (\neg(\neg P \wedge (\neg Q \Rightarrow R)) \vee S) \wedge (\neg S \vee (\neg P \wedge (\neg Q \Rightarrow R))) && (3) \\
\equiv\ & (\neg(\neg P \wedge (\neg\neg Q \vee R)) \vee S) \wedge (\neg S \vee (\neg P \wedge (\neg\neg Q \vee R))) && (3) \\
\equiv\ & ((\neg\neg P \vee (\neg\neg\neg Q \wedge \neg R)) \vee S) \wedge (\neg S \vee (\neg P \wedge (\neg\neg Q \vee R))) && (4) \\
\equiv\ & ((P \vee (\neg Q \wedge \neg R)) \vee S) \wedge (\neg S \vee (\neg P \wedge (Q \vee R))) && (5) \\
\equiv\ & (((P \vee \neg Q) \wedge (P \vee \neg R)) \vee S) \\
& \quad \wedge ((\neg S \vee \neg P) \wedge (\neg S \vee (Q \vee R))) && (6) \\
\equiv\ & (P \vee \neg Q \vee S) \wedge (P \vee \neg R \vee S) \\
& \quad \wedge (\neg S \vee \neg P) \wedge (\neg S \vee Q \vee R) && (6)
\end{aligned}
$$

The result is in conjunctive normal form.

Reduced CNF

There is one more transformation that it is easy and natural to do: simplifying clauses.

A CNF formula is in reduced CNF (RCNF) if, for each of its clauses, no propositional letter occurs twice.

The transformation is the obvious one, for example:

$$(A \vee \neg B \vee \neg A) \wedge (\neg C \vee \neg B) \wedge (C \vee \neg A \vee C \vee B)$$

becomes

$$(\neg C \vee \neg B) \wedge (C \vee \neg A \vee B)$$

Normal Form Does Not Mean Unique Form

The formula

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

is in reduced DNF, and so is the equivalent

$$\neg P \vee Q$$

So two DNF formulas may have different sizes and variables, and yet be equivalent.

Similarly for (R)CNF.

Canonical Forms: Xor Normal Form

If a normal form leads to a unique representation for every Boolean function, we call it canonical.

One canonical form ("xor normal form") presents the function in a sum-of-products form, using exclusive or and conjunction.

For example, the formula $(A \Rightarrow B) \wedge (B \oplus C)$ is written as

$$ABC \oplus AC \oplus B \oplus C$$

This form is unique, up to reordering of conjuncts and summands.

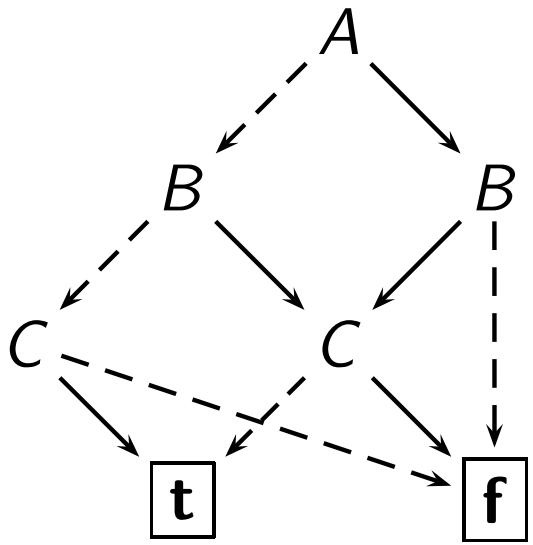
Or, representing the summands as sets:

$$\{\{A, B, C\}, \{A, C\}, \{B\}, \{C\}\}$$

Canonical Forms: ROBDDs

Binary decision diagrams (BDDs) give another canonical form.

Here $(A \Rightarrow B) \wedge (B \oplus C)$ is represented by the graph



Read: If A then [follow solid arc] else [follow dashed arc].

Validity and Satisfiability with ROBDDs

This graph representation becomes canonical when we enforce maximal sharing of subgraphs (and agree on an ordering of variables, like A, B, C).

The resulting graph is then an ROBDD—a reduced, ordered BDD.

These have been very popular and useful for hardware verification etc.

Clearly a propositional formula is valid iff its ROBDD is the single-leaf graph $\boxed{\mathbf{t}}$.

It is unsatisfiable iff its ROBDD is $\boxed{\mathbf{f}}$.

CNF and Clausal Form

Knowledge bases are often presented in CNF, as a set (conjunction) of clauses.

A clause is a set (disjunction) of literals.

Abstracting from the order of literals and clauses, we can think of a formula in CNF as given in clausal form, for example, we may write

$$(P \vee \neg Q \vee S) \wedge (P \vee \neg R \vee S) \wedge (\neg S \vee \neg P) \wedge (\neg S \vee Q \vee R)$$

as

$$\{\{P, S, \neg Q\}, \{P, S, \neg R\}, \{\neg P, \neg S\}, \{Q, R, \neg S\}\}$$

We shall often make no distinction between these.

Empty Clauses

Clause $\{A, B\}$ represents $A \vee B$, and clause $\{A\}$ represents A.

How shall we read the clause $\emptyset$?

The natural reading is that it is false, because $\mathbf{f}$ is neutral element for $\vee$—we could have written $A \vee B$ as $\mathbf{f} \vee A \vee B$, and A as $\mathbf{f} \vee A$.

Hence we agree that the empty clause $\emptyset$ represents $\perp$.

Empty CNF Formulas

The formula $\{C_1, C_2\}$, with clauses C_1 and C_2, represents $C_1 \wedge C_2$.

The formula $\{C\}$ represents C.

How shall we read the (CNF) formula $\emptyset$?

The natural reading is that it is true, because $\mathbf{t}$ is neutral element for $\wedge$—we could have written $C_1 \wedge C_2$ as $\mathbf{t} \wedge C_1 \wedge C_2$, and C as $\mathbf{t} \wedge C$.

Hence we agree that the empty formula $\emptyset$ represents $\top$.

Empty Clauses and Formulas

For clausal form representation (CNF) we then have:

- The set $\emptyset$ of clauses is valid.
- Any set $\{\emptyset, \ldots\}$ of clauses is unsatisfiable.

Don't be confused by this.

An empty set of clauses is valid, because it is trivial to satisfy all of the set's clauses—there is nothing to do.

But a (non-empty) set that contains an empty clause cannot be satisfied, because nothing satisfies that empty clause.

In particular, note that $\{\emptyset\} \neq \emptyset$.

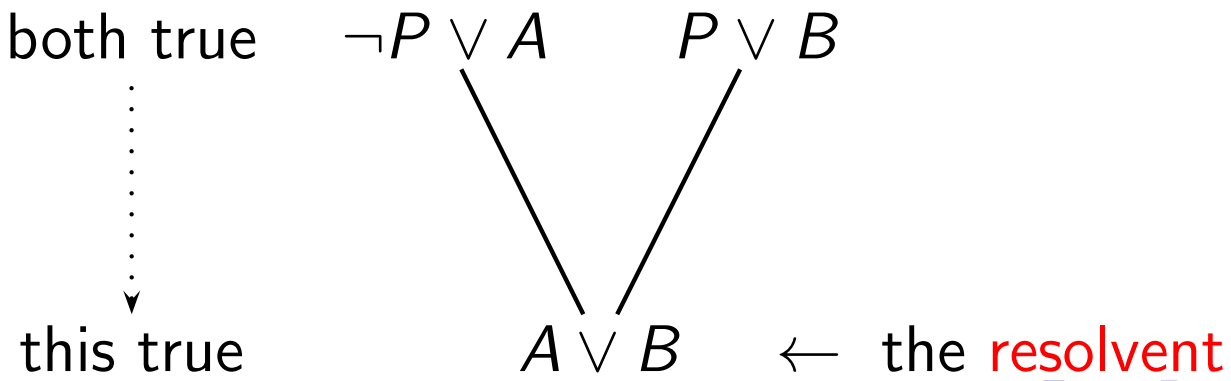
Resolution-Based Inference

Consider the two clauses $\neg P \vee A$ and $P \vee B$

- If P is true, they reduce to A and **t**.
- If P is false, they reduce to **t** and B.

There are no other possible values for P, so: if $(\neg P \vee A)$ and $(P \vee B)$ are both true, then either A is true or B is true (or both). That is, the clause $A \vee B$ is a logical consequence of the two original clauses.

We call $A \vee B$ their resolvent.



Propositional Resolution Generally

Let C_1 and C_2 be clauses such that $P \in C_1$ and $\neg P \in C_2$.

$(C_1 \setminus \{P\}) \cup (C_2 \setminus \{\neg P\})$ is a resolvent of C_1 and C_2.

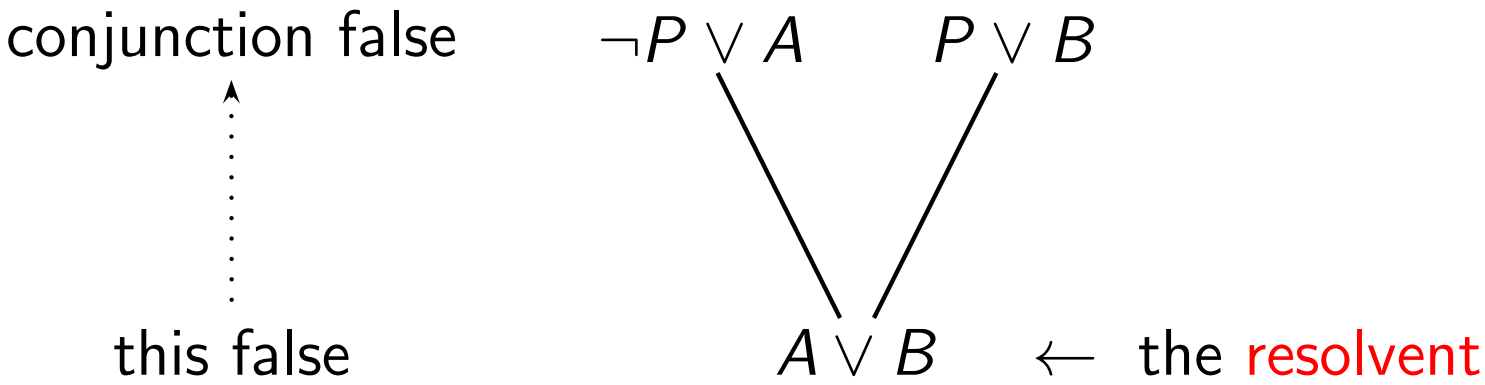
(Note: $A \setminus B$ is set difference: the set of elements in A but not in B.)

Theorem. If R is a resolvent of C_1 and C_2 then $C_1 \wedge C_2 \models R$.

This generalises the well-known inference rule of modus ponens:
From A and $A \Rightarrow B$ deduce B.

Refuting a Set of Clauses

Resolution suggests a way of verifying that a CNF formula is unsatisfiable.



If, through a number of resolution steps, we can derive the empty clause $\perp$, then the original set of clauses were unsatisfiable.

We talk about a refutation proof.

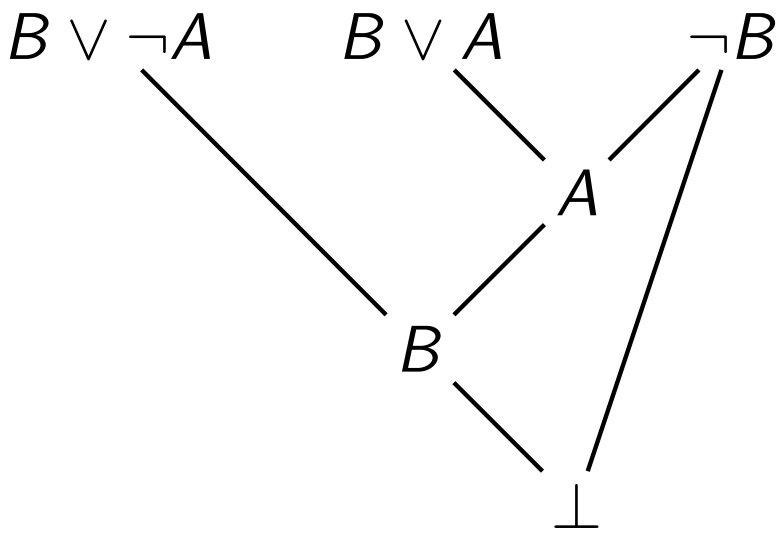
Deductions and Refutations

A resolution deduction of clause C from a set S of clauses is a finite sequence $C_1, C_2, \ldots, C_n$ of clauses such that $C_n = C$ and for each i, $1 \leq i \leq n$, C_i is either

- a member of S, or
- a resolvent of C_j and C_k, for some $j, k < i$.

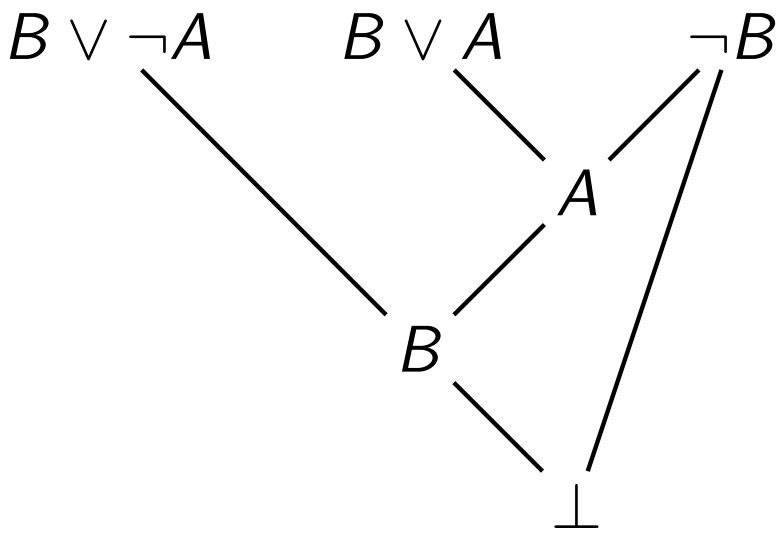
A resolution refutation of a set S of clauses is a resolution deduction of $\perp$ from S.

An Example of a Refutation



This shows that $(B \vee \neg A) \wedge (B \vee A) \wedge \neg B$ is unsatisfiable.

An Example of a Refutation



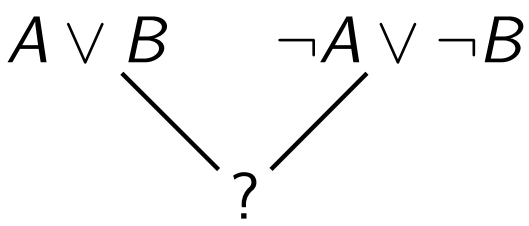
This shows that $(B \vee \neg A) \wedge (B \vee A) \wedge \neg B$ is unsatisfiable.

Exercise:
Find a simpler refutation to show the formula is a contradiction.

Refutation Exercise

Seek a resolution refutation of

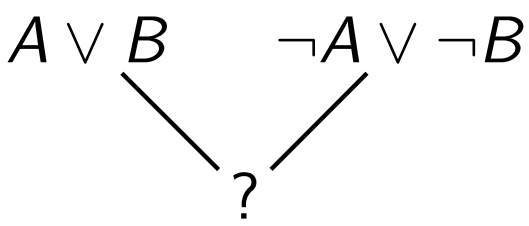
$$(A \vee B) \wedge (\neg A \vee \neg B)$$



Refutation Exercise

Seek a resolution refutation of

$$(A \vee B) \wedge (\neg A \vee \neg B)$$



Not possible to derive $\bot$ here
—which is fortunate, because the original formula is satisfiable.

Note in particular that we cannot "cancel out" several literals in one go.

How to Use Refutations (1)

To show that F is valid, first put $\neg F$ in RCNF, yielding a set S of clauses.

Then refute S, that is, deduce $\bot$ from S.

Consider: $(P \Rightarrow R) \vee (R \Rightarrow (P \wedge Q))$

Negating yields: $\neg((\neg P \vee R) \vee (\neg R \vee (P \wedge Q)))$

Pushing negation then yields: $P \wedge \neg R \wedge R \wedge (\neg P \vee \neg Q)$

From this we can derive $\bot$ in a single resolution step.

How to Use Refutations (2)

Suppose we express a circuit design as a formula F in RCNF.

Suppose we wish to show that the design satisfies some property G, that is, show $F \models G$.

We can exploit this fact: $\boxed{F \models G \text{ iff } F \wedge \neg G \text{ is unsatisfiable}}$

Hence a strategy is:

1. Negate G and bring it into RCNF;
2. add those clauses to the set F; and
3. find a refutation of the resulting set of clauses.

After the Break

We move on to predicate logic.

Grok Worksheet 1 (assessed) opened yesterday. It has a deadline of 30 August. Remember: No extensions are possible.