RESOLUTION

Most useful proof method in implementations.

Proof method by refutation :
as with semantic tableaux, instead of proving A is valid,
we prove $\neg A$ is inconsistent ;
instead of proving $E \models A$
we prove $E \cup \{\neg A\}$ is inconsistent.

Classical resolution requires formulas in clausal form,
or conjunctive normal form.

Normal forms

The expression $(x^2 - 4x)(x + 3) + (2x - 1)^2 + 4x - 19$
is a polynomial, but its properties are not obvious. A more convenient form for the same polynomial will emphasise its degree, its roots, ...
Normal forms (or canonical forms) are used for that purpose. The most used forms are :

$x^3 + 3x^2 - 12x - 18$ (sum of monomials, decreasing degrees) ;

$(x - 3)(x + 3 - \sqrt{3})(x + 3 + \sqrt{3})$ (product of linear factors) ;

$[(x + 3)x - 12]x - 18$ (Horner form).

Disjunctive normal form I

p	q	r	$p \Rightarrow q$	$(p \Rightarrow q) \Rightarrow r$
T	T	T	T	T
T	T	F	T	F
T	F	T	F	T
T	F	F	F	T
F	T	T	T	T
F	T	F	T	F
F	F	T	T	T
F	F	F	T	F

$$\begin{array}{ll} & (\ p \wedge \ q \wedge \ r) \\ \vee & (\ p \wedge \neg q \wedge \ r) \\ \vee & (\ p \wedge \neg q \wedge \neg r) \\ \vee & (\neg p \wedge \ q \wedge \ r) \\ \vee & (\neg p \wedge \neg q \wedge \ r) \end{array}$$

The truthtable of $(p \Rightarrow q) \Rightarrow r$ (left), demonstrates that this formula is logically equivalent to the disjunctive formula (right). Each disjunct corresponds to a "true" line of the table.

A *disjunctive normal form* is a disjunction of *cubes*, which are conjunctions of literals. Every formula has a truthtable and is therefore logically equivalent to a disjunctive normal form (DNF).

Comment. A DNF can contain any (finite) number of cubes ; a cube can contain any (finite) number of literals.

Disjunctive normal form II

The cube $(\ell_1 \wedge \ell_2 \wedge \cdots \wedge \ell_n)\,, \quad (n \in \mathbf{N})$, is sometimes written $\bigwedge\{\ell_1, \ldots, \ell_n\}$, or $\bigwedge_i \ell_i$, or simply $\{\ell_1, \ldots, \ell_n\}$.

Comment. true et *false* are not literals, but they are cubes.

A cube is inconsistent if and only if it contains a pair of opposite literals, a complementary pair.

A cube is valid if and only if it is empty.

A DNF is inconsistent if and only if all its cubes are inconsistent. The empty DNF is therefore inconsistent.

Conjunctive normal form I

A *clause* is a disjunction of literals.

A clause can be represented as $\bigvee\{\ell_i : i = 1, \ldots, n\}$, and even as $\{\ell_i : i = 1, \ldots, n\}$, although the latter is ambiguous and should be avoided.

Comment. Sometimes, a notation like $p\overline{q}r$ is used to denote the cube $p \wedge \neg q \wedge r$ or the clause $p \vee \neg q \vee r$. This is ambiguous and should be avoided.

The only inconsistent clause is the *empty clause*, denoted $\Box$.

A clause is valid if and only if it contains a pair of opposite literals, a complementary pair.

A *unit clause* contains a single literal.

Conjunctive normal form II

A *conjunctive normal form* or CNF is a conjunction of clauses.

Examples :

— $(\neg p \vee q \vee r) \wedge (\neg q \vee r) \wedge (\neg r)$ – CNF

— $(\neg p \vee q \vee r) \wedge \neg(\neg q \vee r) \wedge (\neg r)$ – not CNF

A CNF is valid if and only if all its clauses are valid ; as a consequence, the empty CNF is valid.

Every formula is logically equivalent to some CNF.

Comment. Clauses, cubes, DNF and CNF are formulas and therefore contain finitely many terms.

Why normal forms ?

A useful normal form must be

- — general enough : any formula should have a logically equivalent normal form ;
- — as specific as possible, so specific algorithms can be designed to deal with normal forms, more efficient than the general algorithms.

Normal forms could be unique, but that is not true for DNF and CNF.

Example. The DNF
$(p \wedge q \wedge r) \vee (p \wedge \neg q \wedge r) \vee (p \wedge \neg q \wedge \neg r) \vee$
$(\neg p \wedge q \wedge r) \vee (\neg p \wedge \neg q \wedge \neg r)$
is logically equivalent to a shorter DNF :
$(p \wedge r) \vee (\neg q \wedge \neg r) \vee (\neg p \wedge q \wedge r)$

Normalization algorithm I

From now on, only CNF is considered.

1. Eliminate all connectives but $\neg$, $\vee$, $\wedge$.
2. Use De Morgan laws for propagating $\neg$ occurrences down the syntactic tree.

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

3. Eliminate double negations.

$$\neg\neg A \leftrightarrow A$$

4. Use distributivity laws to propagate $\vee$ downward.

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

A CNF can be viewed as a set of clauses, a *clausal form*.

Exercise. Observe the link beteen a CNF logically equivalent to A and a DNF logically equivalent to $\neg A$.

Normalization algorithm II

Variable L is a (conjunctive) set of disjunctions ; its initial value is $\{A\}$ where A is any formula (viewed as a disjunction of one term). The final value of L is a CNF, logically equivalent to A. We call *disclause* any disjunction containing at least one term which is not a literal.

```
$L := \{A\}$ ;
As long as $L$ contains some disclause do
    $\{ \bigwedge L \leftrightarrow A$ is invariant $\}$
    select a disclause $D \in L$ ;
    select a non-literal $t \in D$ ;
    if $t = \alpha$ do
        $t_1 := \alpha_1$; $t_2 := \alpha_2$;
        $D_1 := (D - t) + t_1$; $D_2 := (D - t) + t_2$;
        $\{ D \longleftrightarrow D_1 \wedge D_2 \}$
        $L := (L \backslash \{D\}) \cup \{D_1, D_2\}$
    else ($t = \beta$) do
        $t_1 := \beta_1$; $t_2 := \beta_2$;
        $D' := ((D - t) + t_1) + t_2$;
        $\{ D \longleftrightarrow D' \}$
        $L := (L \backslash \{D\}) \cup \{D'\}$
```

Example

Design a CNF logically equivalent to $(\neg p \Rightarrow \neg q) \Rightarrow (p \Rightarrow q)$.

$(\neg p \Rightarrow \neg q) \Rightarrow (p \Rightarrow q)$

$\neg(\neg\neg p \vee \neg q) \vee (\neg p \vee q)$ ($\Rightarrow$ elimination)

$(\neg\neg\neg p \wedge \neg\neg q) \vee (\neg p \vee q)$ (downward propagation, $\neg$)

$(\neg p \wedge q) \vee (\neg p \vee q)$ (double negation)

$(\neg p \vee \neg p \vee q) \wedge (q \vee \neg p \vee q)$ (distributivity)

Formula $(\neg p \Rightarrow \neg q) \Rightarrow (p \Rightarrow q)$ is logically equivalent to CNF $(\neg p \vee \neg p \vee q) \wedge (q \vee \neg p \vee q)$.

It is also logically equivalent to $\neg p \vee q$.

Example (bis)

1.	$\{(\neg p \Rightarrow \neg q) \Rightarrow (p \Rightarrow q)\}$	$Init$
2.	$\{\neg(\neg p \Rightarrow \neg q) \vee (p \Rightarrow q)\}$	$\beta, 1$
3.	$\{\neg(\neg p \Rightarrow \neg q) \vee \neg p \vee q\}$	$\beta, 2$
4.	$\{\neg p \vee \neg p \vee q\,,\ \neg\neg q \vee \neg p \vee q\}$	$\alpha, 3$
5.	$\{\neg p \vee \neg p \vee q\,,\ q \vee \neg p \vee q\}$	$\alpha, 4$

Therefore the CNF is

$(\neg p \vee \neg p \vee q) \wedge (q \vee \neg p \vee q)\,.$

It can be simplified into

$(\neg p \vee q) \wedge (\neg p \vee q)\,,$

and further into

$\neg p \vee q\,.$

Simplification of clausal forms

The normalization algorithm usually leads to CNF that can (should) be simplified.

1. Keep only one occurrence of a literal inside a clause.

   Example : $(\neg p \vee q \vee \neg p) \wedge (r \vee \neg p) \longleftrightarrow (\neg p \vee q) \wedge (r \vee \neg p)$

2. Valid clauses (containing a complementary pair) can be omitted.

   Example : $(\neg p \vee q \vee p) \wedge (r \vee \neg p) \longleftrightarrow (r \vee \neg p)$

3. If a clause c_1 is included into a clause c_2, then c_2 can be omitted.

   Example : $(r \vee q \vee \neg p) \wedge (\neg p \vee r) \longleftrightarrow (\neg p \vee r)$

These simplifications lead to a *pure* normal form, which is still not unique. For instance, $(p \vee \neg q) \wedge q$ and $p \wedge q$ are pure, logically equivalent CNFs.

Resolution rule I

A clause set (set of clauses) S is inconsistent if and only if $S \models \Box$.
($\Box$ is the empty clause, also denoted *false*.)

Idea. Demonstrate S inconsistency by "deriving" $\Box$ (*false*) from S.

Let A, B, X be formulas, let v be a valuation.

Assume $v(A \vee X) = T$ and $v(B \vee \neg X) = T$.

If $v(X) = T$, then $v(B) = T$,
therefore $v(A \vee B) = T$.

If $v(X) = F$, then $v(A) = T$,
therefore $v(A \vee B) = T$.

As a result, $\{(A \vee X), (B \vee \neg X)\} \models (A \vee B)$.

Resolution rule : special case where X is a proposition and where A, B are clauses.

Resolution rule II

Relation $\vdash_{\mathcal{R}}$ (or $\vdash$) is inductively defined
between a clause set and a clause ;
it is the smallest relation satisfying these conditions :

1. If $C \in S$, then $S \vdash C$.
2. Let $C_1 = (C_1' \vee p)$ and $C_2 = (C_2' \vee \neg p)$;
   if $S \vdash C_1$ and $S \vdash C_2$, then $S \vdash C_1' \vee C_2'$.

Clauses C_1 and C_2 can be *resolved* (with respect to p) ;
clause *Res*$(C_1, C_2) =_{def} C_1' \vee C_2'$ is their *resolvent*.

If S is a clause set, S^R is defined as the smallest superset of S
containing the resolvents of its elements.

$S^R = \{C : S \vdash C\} = \{C : S^R \vdash C\}$.

Soundness of resolution rule

Let S a clause set and C a clause. We must prove, if $S \vdash C$, then $S \models C$. It is sufficient to see that relation $\models$ (restricted to clause sets and clauses) satisfies the characteristic conditions of relation $\vdash_{\mathcal{R}}$:

1. If $C \in S$, then $S \models C$.
2. Let $C_1 = (C_1' \vee p)$ and $C_2 = (C_2' \vee \neg p)$;
   if$S \models C_1$ and $S \models C_2$, then $S \models C_1' \vee C_2'$.

Condition 1 is obviously satisfied ; condition 2 results from $\{(A \vee X), (B \vee \neg X)\} \models (A \vee B)$.

Comment. Clause sets S and S^R are always logically equivalent.

Completeness of resolution rule I

If S is a clause set, if A is a clause and if $S \models A$, can we deduce $S \vdash_{\mathcal{R}} A$?
Obviously not :

$$\{p, \neg p\} \models q \,,$$

but

$$\{p, \neg p\} \not\vdash_{\mathcal{R}} q \,.$$

Fortunately, we do not need so much ; instead of proving $S \models A$, we prove the equivalent $S, \neg A \models \Box$.
The following result can be used :

Theorem. If $S \models \Box$, then $S \vdash_{\mathcal{R}} \Box$.

This “weak completeness” is in fact as powerful as completeness (why ?).

Semantic tree

Let S a formula or a formula set, with $\Pi_S = \{p_1, p_2, \ldots\}$.

A *semantic tree* is a complete, balanced binary tree, labelled as follows : left branches at level i are labelled p_i and right branches are labelled $\neg p_i$.

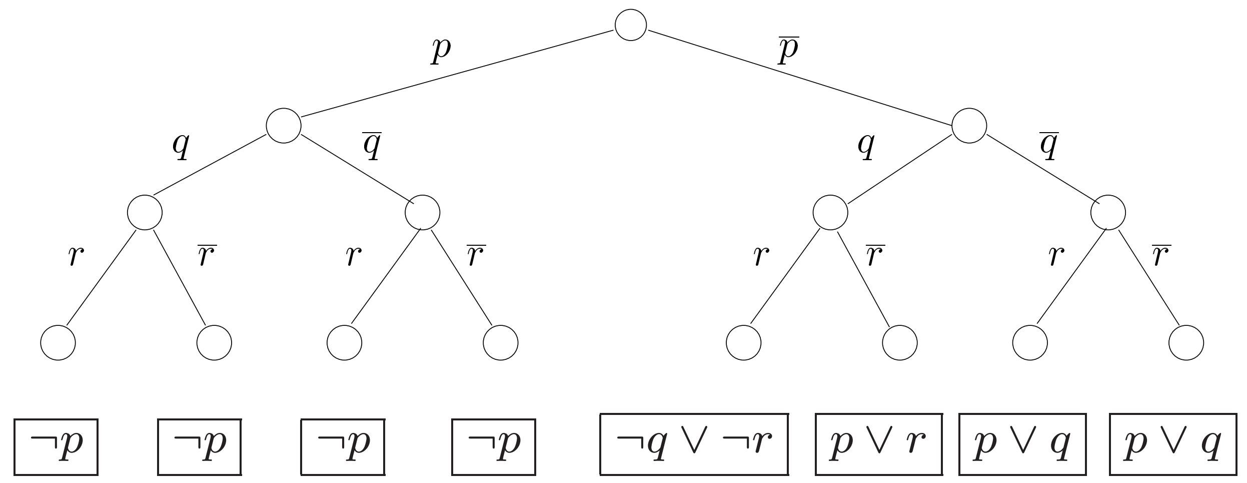
The leaves (or full branches) of the semantic tree S correspond to the valuations of Π_S and S.

Each path $\mathcal{C}$ from the root to some node n at level i defines

— a proposition set, $\Pi(n) = \{p_1, \ldots, p_i\}$;
— a valuation v_n on this set ;
$v_n(p_k) = T$ if $p_k \in \mathcal{C}$ and $v_n(p_k) = F$ if $\neg p_k \in \mathcal{C}$.

Semantic tree : an example

Let $S = \{p \vee q, p \vee r, \neg q \vee \neg r, \neg p\}$, a clause set.
$\Pi_S = \{p, q, r\}$.
A semantic tree is :



The tree is finite since Π_S is finite.

As S is inconsistent, each leaf can be labelled with a clause made false by the valuation associated with that leaf.

Completeness of the resolution method (finite case) I

If S is a finite inconsistent clause set, then $S \vdash \Box$.

Let $\mathcal{A}$ be a semantic tree for S.

The path from the root to node n defines a proposition set $\Pi(n)$ and a valuation v_n for this set ; $v_n(\ell) = T$ for each labelling literal ℓ on the path.

S is inconsistent, so the valuation associated with any leaf f of $\mathcal{A}$ falsifies some clause $C_f \in S$. We label f with C_f. Observe that

$$\Pi_{C_f} \subseteq \Pi(f) = \Pi_S \quad \text{et} \quad v_f(C_f) = F\,.$$

(Π_{C_f} is the set of atoms occurring in C_f.)
We will attempt to propagate leaf labelling upward : each node n will be labelled with some clause $C_n \in S^R$ such that

$$\Pi_{C_n} \subseteq \Pi(n) \subseteq \Pi_S \quad \text{and} \quad v_n(C_n) = F.$$

If this propagation succeeds, root r will be labelled with $C_r \in S^R$ such that

$$\Pi_{C_r} \subseteq \Pi(r) \quad \text{et} \quad v_r(C_r) = F.$$

As $\Pi(r) = \emptyset$ and $v_r(C_r) = F$, the only possibility is $C_r = \Box$.

Completeness of the resolution method (finite case) II

How to label node n ?

Let n_1, n_2 the children of node n ; assume

$$\Pi(n_1) = \Pi(n_2) = \Pi(n) \cup \{p\}.$$

$$\begin{tikzcd}
 & n(C_n) \arrow[dl, "p"'] \arrow[dr, "\neg p"] & \\
n_1(C_{n_1}) & & n_2(C_{n_2})
\end{tikzcd}$$

Assume

$C_{n_1} \in S^R$ and $\Pi_{C_{n_1}} \subseteq \Pi(n_1)$ and $v_{n_1}(C_{n_1}) = F$
$C_{n_2} \in S^R$ and $\Pi_{C_{n_2}} \subseteq \Pi(n_2)$ and $v_{n_2}(C_{n_2}) = F$

Node n is labelled as follows :

— If $p \notin \Pi_{C_{n_i}}$ for $i = 1$ or 2, then $C_n = C_{n_i}$.
— If $p \in \Pi_{C_{n_1}}$ and $p \in \Pi_{C_{n_2}}$:
$v_{n_1}(C_{n_1}) = F$ so $C_{n_1} = C'_{n_1} \vee \neg p$ and $v_{n_2}(C_{n_2}) = F$ so $C_{n_2} = C'_{n_2} \vee p$.
Let $C_n = C'_{n_1} \vee C'_{n_2}$ ($= \mathit{Res}_p(C_{n_1}, C_{n_2})$).

In both cases : $C_n \in S^R$ and $\Pi_{C_n} \subseteq \Pi(n)$ et $v_n(C_n) = F$.

and the completeness (finite case) is proved.

Completeness of the resolution method (infinite case)

Due to the compactness theorem, the statement

$$\Box \in S^R \quad \text{iff} \quad S \text{ is inconsistent}$$

remains true if S is infinite.

If $\Box \in S^R$, then S^R and therefore S are inconsistent.

If S is inconsistent, there is a finite inconsistent subset S_f, so $\Box \in S_f^R$ and therefore $\Box \in S^R$ since $S_f^R \subset S^R$.

Resolution procedure I

If S is a clause set,
let $\mathcal{M}_S$ be the set of all models of S.
S is inconsistent iff $\mathcal{M}_S = \emptyset$.

Resolution procedure

```
$S := S_0$ ;     ($S_0$ clause set)
$\{\mathcal{M}_S = \mathcal{M}_{S_0}\}$
While $\Box \notin S$, do :
    select $p \in \Pi_S$,
            $C_1 = (C_1' \vee p) \in S$,
            $C_2 = (C_2' \vee \neg p) \in S$ ;
    $S := S \cup \{\mathit{Res}(C_1, C_2)\}$
    $\{\mathcal{M}_S = \mathcal{M}_{S_0}\}$
```

Comment on selection procedure : each resolvent pair can be selected only once ; this provides termination since, if the lexicon size is n, no more than 3^n (non valid) clauses can be generated.

Resolution procedure II

Invariant : only logical consequences are inserted into S so the set $\mathcal{M}_S$ does not change.

The procedure terminates smoothly (false guard) or aborts (no possible selection).

Smooth termination : when the guard is false and the computation stops, the final value S_f is such that $\mathcal{M}_{S_f} = \mathcal{M}_{S_0}$ and $\Box \in S_f$, so S_f and S_0 are inconsistent.

Abortion : If all resolvents have been produced and none of them is $\Box$, then $\mathcal{M}_{S_f} = \mathcal{M}_{S_0}$ and $\Box \notin S_f$; both S_f and S_0 are consistent.

A derivation of $\Box$ (*false*) from S is a *refutation* of S.

Refutations : examples I

Let $S = \{(p \vee q), (p \vee r), (\neg q \vee \neg r), (\neg p)\}$.

Clause numbering :

1.	$p \vee q$
2.	$p \vee r$
3.	$\neg q \vee \neg r$
4.	$\neg p$

Two refutations :

5.	$p \vee \neg r$	$(1, 3)$
6.	q	$(1, 4)$
7.	$p \vee \neg q$	$(2, 3)$
8.	r	$(2, 4)$
9.	p	$(2, 5)$
10.	$\neg r$	$(3, 6)$
11.	$\neg q$	$(3, 8)$
12.	$\neg r$	$(4, 5)$
13.	$\neg q$	$(4, 7)$
14.	$\square$	$(4, 9)$

5.	q	$(1, 4)$
6.	r	$(2, 4)$
7.	$\neg q$	$(3, 6)$
8.	$\square$	$(5, 7)$

Refutations : examples II

Let $S = \{p, \neg p \vee q\}$.
Clause numbering :

1. p
2. $\neg p \vee q$

Derivation :

3. q $(1, 2)$

Let $S = \{p, \neg p \vee q, \neg q\}$.
Clause numbering :

1. p
2. $\neg p \vee q$
3. $\neg q$

Refutation :

4. q $(1, 2)$
5. $\Box$ $(3, 4)$