Little Engines of Proof

N. Shankar, L. de Moura, H. Ruess, A. Tiwari

shankar@csl.sri.com

URL: http://www.csl.sri.com/~shankar/LEP.html

Computer Science Laboratory

SRI International

Menlo Park, CA

Refutation Decision Procedures

A decision procedure determines if a collection of formulas is satisfiable.

A decision procedure is given by a collection of reduction rules on a *logical state* ψ.

State ψ is of the form $\kappa_1|\ldots|\kappa_n$, where each κ_i is a *configuration*.

The logical content of κ is either $\perp$ or is given by a finite set of formulas of the form $A_1, \ldots, A_m$.

A state ψ of the form $\kappa_1, \ldots, \kappa_n$ is satisfiable if some configuration κ_i is satisfiable.

A configuration κ of the form $A_1, \ldots, A_m$ is satisfiable if there is an interpretation M and an assignment ρ such that $M, \rho \models A_i$ for $1 \leq i \leq m$.

Inference Systems for Decision Procedures

A refutation procedure proves A by refuting $\neg A$ through the application of reduction rules.

An application of an reduction rule transforms a state ψ to a state ψ' (written $\psi \Rrightarrow \psi'$).

The states ψ and ψ' must be *equivalent*: Any $\mathcal{M}, \rho$ such that $\mathcal{M}, \rho \models \psi$ there is a ρ' extending ρ such that $\mathcal{M}, \rho' \models \psi'$, and conversely whenever $\mathcal{M}, \rho' \models \psi'$, there is a ρ extending ρ' such that $\mathcal{M}, \rho' \models \psi$.

If relation $\Rrightarrow$ between states is well-founded and any non-bottom irreducible state is satisfiable, we say that the inference system is a decision procedure.

Ex: Prove that a decision procedure as given above is sound and complete.

Truth Table

An inference rule $\dfrac{\kappa}{\kappa_1|\ldots|\kappa_n}$ is shorthand for $\dfrac{\psi[\kappa]}{\psi[\kappa_1|\ldots|\kappa_n]}$.

The *truth table* procedure can be viewed as a *model elimination* procedure.

$$\frac{\Gamma}{\Gamma, p \mid \Gamma, \neg p}\,split \qquad p \text{ and } \neg p \text{ are not in } \Gamma.$$

$$\frac{\Gamma, F}{\perp}\,elim \qquad F \text{ is falsified by the literals in } \Gamma.$$

A *literal* is a *proposition* or the *negation of a proposition*.

The literals in Γ can be viewed as a *partial interpretation*.

Ex: Prove correctness (soundness, termination, and completeness).

Truth Table (Example)

A truth table refutation of $\{p \vee \neg q \vee \neg r, p \vee r, p \vee q, \neg p\}$:

$$\begin{array}{c}
p \vee \neg q \vee \neg r, p \vee r, p \vee q, \neg p \\ \hline
p \vee \neg q \vee \neg r, p \vee r, p \vee q, \neg p, q \mid p \vee \neg q \vee \neg r, p \vee r, p \vee q, \neg p, \neg q \\ \hline
p \vee \neg q \vee \neg r, p \vee r, p \vee q, \neg p, q \mid \bot \\ \hline
p \vee \neg q \vee \neg r, p \vee r, p \vee q, \neg p, q \\ \hline
p \vee \neg q \vee \neg r, p \vee r, p \vee q, \neg p, q, r \mid p \vee \neg q \vee \neg r, p \vee r, p \vee q, \neg p, q, \neg r \\ \hline
\bot \mid p \vee \neg q \vee \neg r, p \vee r, p \vee q, \neg p, q, \neg r \\ \hline
p \vee \neg q \vee \neg r, p \vee r, p \vee q, \neg p, q, \neg r \\ \hline
\bot
\end{array}$$

Ex: Implement the truth table procedure.

Semantic Tableaux

The inference rules for the Semantic Tableaux procedure are:

$\dfrac{A \wedge B, \Gamma}{A, B, \Gamma} \wedge +$	$\dfrac{\neg(A \wedge B), \Gamma}{\neg A, \Gamma \mid \neg B, \Gamma} \wedge -$
$\dfrac{\neg(A \vee B), \Gamma}{\neg A, \neg B, \Gamma} \vee -$	$\dfrac{(A \vee B), \Gamma}{A, \Gamma \mid B, \Gamma} \vee +$
$\dfrac{\neg(A \Rightarrow B), \Gamma}{A, \neg B, \Gamma} \Rightarrow -$	$\dfrac{(A \Rightarrow B), \Gamma}{\neg A, \Gamma \mid B, \Gamma} \Rightarrow +$
$\dfrac{\neg\neg A, \Gamma}{A, \Gamma} \neg$	$\dfrac{A, \neg A, \Gamma}{\bot} \bot$

Semantic Tableaux is a *"DNF translator"*.

Ex: Prove correctness.

Semantic Tableaux (Example)

Refutation of $\neg(p \vee q \Rightarrow q \vee p)$:

$$\begin{array}{c}
\neg(p \vee q \Rightarrow q \vee p) \\ \hline
p \vee q, \neg(q \vee p) \\ \hline
p, \neg(q \vee p) \mid q, \neg(q \vee p) \\ \hline
p, \neg q, \neg p \mid q, \neg(q \vee p) \\ \hline
\bot \mid q, \neg(q \vee p) \\ \hline
q, \neg(q \vee p) \\ \hline
q, \neg q, \neg p \\ \hline
\bot
\end{array}$$

Ex: Use the Semantic Tableaux procedure to refute $\neg(p \vee (q \wedge r) \Rightarrow (p \vee q) \wedge (p \vee r))$.

Ex: Implement the Semantic Tableaux.

Semantic Tableaux (Cont.)

The complexity of *Semantic Tableaux* proofs depends on the *length* of the *formula* to be decided.

The complexity of the *truth-table* procedure depends only on the number of *distinct propositional variables* which occur in it.

The *Semantic Tableaux* procedure does not *p-simulate* the *truth-table* procedure. Consider *fat* formulas such as:

$$\begin{array}{c}
(p_1 \vee p_2 \vee p_3) \wedge (\neg p_1 \vee p_2 \vee p_3) \wedge \\
(p_1 \vee \neg p_2 \vee p_3) \wedge (\neg p_1 \vee \neg p_2 \vee p_3) \wedge \\
(p_1 \vee p_2 \vee \neg p_3) \wedge (\neg p_1 \vee p_2 \vee \neg p_3) \wedge \\
(p_1 \vee \neg p_2 \vee \neg p_3) \wedge (\neg p_1 \vee \neg p_2 \vee \neg p_3)
\end{array}$$

Ex: Use Semantic Tableaux to refute the formula above.

Semantic Tableaux (Cont.)

The classical notion of truth is governed by two basic principles:

Non-contradiction no proposition can be true and false at the same time.

Bivalence every proposition is either true of false.

There is *no rule* in the *Semantic Tableaux* procedure which correspondes to the *principle of bivalence*.

The elimination of the *principle of bivalence* seem to be inadequate from the point of view of efficiency.

Semantic Tableaux + Bivalence

The *principle of bivalence* can be recovered if we replace the Semantic Tableaux *branching* rules by:

$\dfrac{\neg(A \wedge B), \Gamma}{\neg A, \Gamma \mid A, \neg B, \Gamma} \wedge_{left}-$	$\dfrac{\neg(A \wedge B), \Gamma}{\neg B, \Gamma \mid B, \neg A, \Gamma} \wedge_{right}-$
$\dfrac{(A \vee B), \Gamma}{A, \Gamma \mid \neg A, B, \Gamma} \vee_{left}+$	$\dfrac{(A \vee B), \Gamma}{B, \Gamma \mid \neg B, A, \Gamma} \vee_{right}+$
$\dfrac{(A \Rightarrow B), \Gamma}{\neg A, \Gamma \mid A, B, \Gamma} \Rightarrow_{left}+$	$\dfrac{(A \Rightarrow B), \Gamma}{B, \Gamma \mid \neg B, \neg A, \Gamma} \Rightarrow_{right}+$

The new rules are *asymmetric*.

Ex: Show that the new rules are sound.

CNF

A *CNF* formula is a conjunction of *clauses*. A *clause* is a disjunction of *literals*.

Ex: Implement a linear-time decision procedure for 2CNF (each clause has at most 2 literals).

A clause is *trivial* if it contains a *complementary* pair of literals.

Since the *order* of the *literals* in a clause is *irrelevant*, the clause can be treated as a *set*.

A set of clauses is *trivial* if it contains the *empty clause* (false).

CNF (cont.)

Equivalence rules can be used to translate any formula to CNF.

eliminate $\Rightarrow$	$A \Rightarrow B \equiv \neg A \vee B$
reduce the scope of $\neg$	$\neg(A \vee B) \equiv \neg A \wedge \neg B$, $\neg(A \wedge B) \equiv \neg A \vee \neg B$
apply distributivity	$A \vee (B \wedge C) \equiv (A \vee B) \wedge (A \vee C)$, $A \wedge (B \vee C) \equiv (A \wedge B) \vee (A \wedge C)$

CNF (cont.)

The CNF translation described in the previous slide is too *expensive* (distributivity rule).

However, there is a *linear time* translation to CNF that produces an *equisatisfiable* formula. Replace the distributivity rules by the following rules:

$$\frac{F[l_i \ op \ l_j]}{F[x], x \Leftrightarrow l_i \ op \ l_j} *$$

$$\frac{x \Leftrightarrow l_i \vee l_j}{\neg x \vee l_i \vee l_j, \neg l_i \vee x, \neg l_j \vee x}$$

$$\frac{x \Leftrightarrow l_i \wedge l_j}{\neg x \vee l_i, \neg x \vee l_j, \neg l_i \vee \neg l_j \vee x}$$

(*) x must be a fresh variable.

Ex: Show that the rules preserve equisatisfiability.

CNF translation (example)

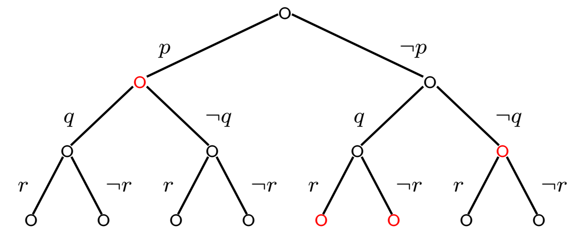
Translation of $(p \wedge (q \vee r)) \vee t$:

$$\begin{array}{c}
(p \wedge (q \vee r)) \vee t \\ \hline
(p \wedge x_1) \vee t, x_1 \Leftrightarrow q \vee r \\ \hline
x_2 \vee t, x_2 \Leftrightarrow p \wedge x_1, x_1 \Leftrightarrow q \vee r \\ \hline
x_2 \vee t, \neg x_2 \vee p, \neg x_2 \vee x_1, \neg p \vee \neg x_1 \vee x_2, x_1 \Leftrightarrow q \vee r \\ \hline
x_2 \vee t, \neg x_2 \vee p, \neg x_2 \vee x_1, \neg p \vee \neg x_1 \vee x_2, \neg x_1 \vee q \vee r, \neg q \vee x_1, \neg r \vee x_1
\end{array}$$

Ex: Implement a CNF translator.

Semantic Trees

A *semantic tree* represents the set of partial interpretations for a set of clauses. A semantic tree for $\{p \vee \neg q \vee \neg r, p \vee r, p \vee q, \neg p\}$:



A node N is a failure node if its associated interpretation *falsifies* a clause, but its ancestor doesn't.

Ex: Show that the semantic tree for an unsatisfiable (non-trivial) set of clauses must contain a non failure node such that its descendants are failure nodes.

Resolution

Formula must be in *CNF*.

Resolution procedure uses only *one rule*:

$$\frac{C_1 \vee p, C_2 \vee \neg p}{C_1 \vee p, C_2 \vee \neg p, C_1 \vee C_2} res$$

The result of the resolution rule is also a clause, it is called the *resolvent*. *Duplicate literals* in a clause and *trivial clauses* are *eliminated*.

There is no *branching* in the resolution procedure.

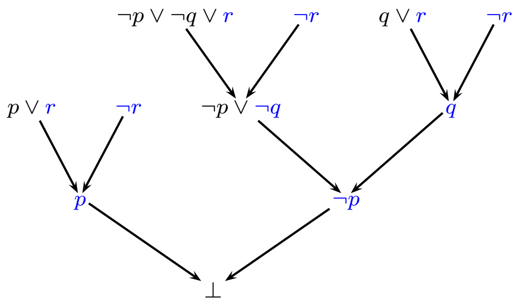
Example: The resolvent of $p \vee q \vee r$, and $\neg p \vee r \vee t$ is $q \vee r \vee t$.

Termination argument: there is a *finite* number of distinct clauses over n propositional variables.

Ex: Show that the resolution rule is sound.

Resolution (example)

A refutation of $\neg p \vee \neg q \vee r$, $p \vee r$, $q \vee r$, $\neg r$:



Ex: Implement a naïve resolution procedure.

Completeness of the Resolution Procedure

Let $Res(S)$ be the closure of S under the resolution rule.

Completeness: S is unsatisfiable iff $Res(S)$ contains the *empty clause*.

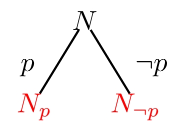
Proof ($\Rightarrow$):

Assume that S is unsatisfiable, and $Res(S)$ does not contain the *empty clause*.

Key points: $Res(S)$ is unsatisfiable, and $Res(S)$ is a non trivial set of clauses.

The semantic tree of $Res(S)$ must contain a non failure node N such that its descendants (N_p, $N_{\neg p}$) are failure nodes.

Completeness of the Resolution Procedure (cont.)



There is $C_1 \vee \neg p$ which is falsified by N_p, but not by N.

There is $C_2 \vee p$ which is falsified by $N_{\neg p}$, but not by N.

$C_1 \vee C_2$ is the resolvent of $C_1 \vee \neg p$ and $C_2 \vee p$.

$C_1 \vee C_2$ is in $Res(S)$, and it is falsified by N (*contradiction*).

Proof ($\Leftarrow$): $Res(S)$ is unsatisfiable, and equivalent to S. So, S is unsatisfiable.

Subsumption

The *resolution* procedure may generate several *irrelevant* and *redundant clauses*.

Subsumption is a clause *deletion strategy* for the resolution procedure.

$$\frac{C_1, C_1 \vee C_2}{C_1}\,sub$$

Example: $p \vee \neg q$ subsumes $p \vee \neg q \vee r \vee t$.

Deletion strategy: Remove the subsumed clauses.

Unit & Input Resolution

Unit resolution: one of the clauses is a unit clause.

$$\frac{C \vee \bar{l}, l}{C, l}\,unit$$

Unit resolution always *decreases* the configuration *size* ($C \vee \bar{l}$ is subsumed by C).

Input resolution: one of the clauses is in S.

Ex: Show that the unit and input resolution procedures are not complete.

Ex: Show that a set of clauses S has an unit refutation iff it has an input refutation (hint: induction on the number of propositions).

Horn Clauses

Each clause has at most on positive literal.

Rule base systems ($\neg p_1 \vee \ldots \vee \neg p_n \vee q \ \equiv\ p_1 \wedge \ldots \wedge p_n \Rightarrow q$).

Positive unit rule:

$$\frac{C \vee \neg p, p}{C, p}\,unit^{+}$$

Horn clauses are the basis of programming languages as *Prolog*.

Ex: Show that the positive unit rule is a complete procedure for Horn clauses.

Ex: Implement a linear time algorithm for Horn clauses.

Semantic Resolution

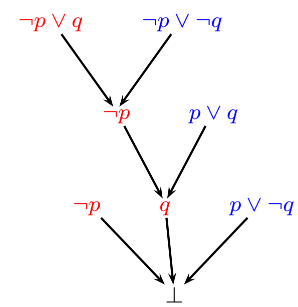
Remark: An interpretation I can be used to *divide* an *unsatisfiable* set of clauses S.

Let I be an *interpretation*, and P an ordering on the propositional variables. A finite set of clauses $\{E_1, \ldots, E_q, N\}$ is called a *clash* with respect to P and I, if and only if:

- E_1, ..., E_q are *false* in I.
- $R_1 = N$, for each $i = 1, \ldots, q$, there is a resolvent R_{i+1} of R_i and E_i.
- The literal in E_i, which is resolved upon, contains the *largest* propositional variable.
- R_{q+1} is *false* in I. R_{q+1} is the *PI-resolvent* of the *clash*.

Semantic Resolution (example)

Let $I = \{p, \neg q\}$, $S = \{p \vee q, \neg p \vee q, p \vee \neg q, \neg p \vee \neg q\}$, and $P = [p < q]$.



Ex: Show that *PI-resolution* is complete (hint: induction on the number of propositions).

Special cases of Semantic Resolution

Positive Hyperresolution: I contains only *negative literals*.

Negative Hyperresolution: I contains only *positive literals*.

A subset T of a set of clauses S is called a *set-of-support* of S if $S - T$ is satisfiable.

A *set-of-support resolution* is a resolution of two clauses that are not both from $S - T$.

Ex: Show that set-of-support resolution is complete (hint: use PI-resolution completeness).

Basic Davis Putnam

Davis Putnam = Unit resolution + Split rule.

$$\frac{\Gamma}{\Gamma, p \mid \Gamma, \neg p} split \quad p \text{ and } \neg p \text{ are not in } \Gamma.$$

$$\frac{C \vee \bar{l}, l}{C, l} unit$$

Used in the most efficient SAT solvers.

Next lecture, we will describe several refinements of the Davis Putnam procedure.