AAA615: Formal Methods

Lecture 1 — Propositional Logic

Hakjoo Oh
2017 Fall

Syntax

- **Atom**: basic elements
  - truth symbols $\bot$ ("false") and $\top$ ("true")
  - propositional variables $P, Q, R, \ldots$
- **Literal**: an atom α or its negation $\neg\alpha$.
- **Formula**: a literal or the application of a logical connective (boolean connective) to formulas

$$
\begin{array}{lcll}
F & \to & \bot & \\
 & | & \top & \\
 & | & P & \\
 & | & \neg F & \text{negation ("not")} \\
 & | & F_1 \land F_2 & \text{conjunction ("and")} \\
 & | & F_1 \lor F_2 & \text{disjunction ("or")} \\
 & | & F_1 \to F_2 & \text{implication ("implies")} \\
 & | & F_1 \leftrightarrow F_2 & \text{iff ("if and only if")}
\end{array}
$$

Syntax

- Formula G is a **subformula** of formula F if it occurs syntactically within G.

$$\begin{array}{rcl} \textbf{sub}(\bot) & = & \{\bot\} \\ \textbf{sub}(\top) & = & \{\top\} \\ \textbf{sub}(P) & = & \{P\} \\ \textbf{sub}(\neg F) & = & \{\neg F\} \cup \textbf{sub}(F) \\ \textbf{sub}(F_1 \wedge F_2) & = & \{F_1 \wedge F_2\} \cup \textbf{sub}(F_1) \cup \textbf{sub}(F_2) \end{array}$$

- $F : (P \wedge Q) \rightarrow (P \vee \neg Q)$
  - $\textbf{sub}(F) =$
- The strict subformulas of a formula are all its subformulas except itself.

Semantics

- The semantics of a logic provides its meaning. The meaning of a PL formula is either true or false.
- The semantics of a formula is defined with an **interpretation** that assigns truth values to propositional variables, e.g., $I : \{P \mapsto \textbf{true}, Q \mapsto \textbf{false}, \ldots\}$.
  - $F : P \land Q \to P \lor \neg Q$
- Inductive definition of semantics:
  - We write $I \models F$ if F evaluates to **true** under I.
  - We write $I \not\models F$ if F evaluates to **false** under I.

$$
\begin{array}{ll}
I \models \top, \quad I \not\models \bot, & \\
I \models P & \text{iff} \;\; I[P] = \textbf{true} \\
I \not\models P & \text{iff} \;\; I[P] = \textbf{false} \\
I \models \neg F & \text{iff} \;\; I \not\models F \\
I \models F_1 \land F_2 & \text{iff} \;\; I \models F_1 \text{ and } I \models F_2 \\
I \models F_1 \lor F_2 & \text{iff} \;\; I \models F_1 \text{ or } I \models F_2 \\
I \models F_1 \to F_2 & \text{iff} \;\; I \not\models F_1 \text{ or } I \models F_2 \\
I \models F_1 \leftrightarrow F_2 & \text{iff} \;\; (I \models F_1 \text{ and } I \models F_2) \text{ or } (I \not\models F_1 \text{ and } I \not\models F_2)
\end{array}
$$

Example

Consider the formula

$$F : P \land Q \to P \lor \neg Q$$

and the interpretation

$$I : \{P \mapsto \textbf{true}, Q \mapsto \textbf{false}\}$$

The truth value of F is computed as follows:

1.	$I \models P$	since $I[P] = \textbf{true}$
2.	$I \not\models Q$	since $I[Q] = \textbf{false}$
3.	$I \models \neg Q$	by 2 and semantics of $\neg$
4.	$I \not\models P \land Q$	by 2 and semantics of $\land$
5.	$I \models P \lor \neg Q$	by 1 and semantics of $\lor$
6.	$I \models F$	by 4 and semantics of $\to$

Satisfiability and Validity

- A formula F is **satisfiable** iff there exists an interpretation I such that $I \vDash F$.
- A formula F is **valid** iff for all interpretations I, $I \vDash F$.
- Satisfiability and validity are dual[1]:

$$F \text{ is valid iff } \neg F \text{ is unsatisfiable}$$

- We can check satisfiability by deciding validity, and vice versa.

[1] In logic, functions (or relations) A and B are dual if $A(x) = \neg B(\neg x)$

Deciding Validity and Satisfiability

Two approaches to show F is valid:

- Truth table method performs exhaustive search: e.g., $F : P \land Q \to P \lor \neg Q$.

P	Q	$P \land Q$	$\neg Q$	$P \lor \neg Q$	F
0	0	0	1	1	1
0	1	0	0	0	1
1	0	0	1	1	1
1	1	1	0	1	1

  Impractical and non-applicable to logic with infinite domain (e.g., first-order logic).

- Semantic argument method uses deduction:
  - Assume F is invalid: $I \not\models F$ for some I (falsifying interpretation).
  - Apply deduction rules (proof rules) to derive a contradiction.
  - If every branch of the proof derives a contradiction, then F is valid.
  - If some branch of the proof never derives a contradiction, then F is invalid. This branch describes a falsifying interpretation of F.
- SAT solvers use both search and deduction.

Deduction Rules for Propositional Logic

$$\frac{I \models \neg F}{I \not\models F} \qquad \frac{I \not\models \neg F}{I \models F}$$

$$\frac{I \models F \land G}{I \models F, I \models G} \qquad \frac{I \not\models F \land G}{I \not\models F \mid I \not\models G}$$

$$\frac{I \models F \lor G}{I \models F \mid I \models G} \qquad \frac{I \not\models F \lor G}{I \not\models F, I \not\models G}$$

$$\frac{I \models F \to G}{I \not\models F \mid I \models G} \qquad \frac{I \not\models F \to G}{I \models F, I \not\models G}$$

$$\frac{I \models F \leftrightarrow G}{I \models F \land G \mid I \models \neg F \land \neg G} \qquad \frac{I \not\models F \leftrightarrow G}{I \models F \land \neg G \mid I \models \neg F \land G}$$

$$\frac{I \models F \quad I \not\models F}{I \models \bot}$$

Example 1

To prove that the formula

$$F : P \wedge Q \rightarrow P \vee \neg Q$$

is valid, assume that it is invalid and derive a contradiction:

1.	$I \not\models P \wedge Q \rightarrow P \vee \neg Q$	assumption
2.	$I \models P \wedge Q$	by 1 and semantics of $\rightarrow$
3.	$I \not\models P \vee \neg Q$	by 1 and semantics of $\rightarrow$
4.	$I \models P$	by 2 and semantics of $\wedge$
5.	$I \not\models P$	by 3 and semantics of $\vee$
6.	$I \models \bot$	4 and 5 are contradictory

Example 2

To prove that the formula

$$F : (P \to Q) \land (Q \to R) \to (P \to R)$$

is valid, assume that it is invalid and derive a contradiction:

1.	$I \not\models F$	assumption
2.	$I \models (P \to Q) \land (Q \to R)$	by 1 and semantics of $\to$
3.	$I \not\models P \to R$	by 1 and semantics of $\to$
4.	$I \models P$	by 3 and semantics of $\to$
5.	$I \not\models R$	by 3 and semantics of $\to$
6.	$I \models P \to Q$	2 and semantics of $\land$
7.	$I \models Q \to R$	2 and semantics of $\land$

Two cases consider from 6:

1. $I \not\models P$: contradiction with 4.
2. $I \models Q$: two cases to consider from 7:
   1. $I \not\models Q$: contradiction
   2. $I \models R$: contradiction with 5.

Exercise

Prove that the formula

$$F : P \lor Q \to P \land Q$$

is valid.

Derived Rules

The proof rules are sufficient, but **derived rules** can make proofs more concise. E.g., the rule of modus ponens:

$$\frac{I \models F \qquad I \models F \rightarrow G}{I \models G}$$

The proof of the validity of the formula:

$$F : (P \rightarrow Q) \land (Q \rightarrow R) \rightarrow (P \rightarrow R)$$

1.	$I \not\models F$	assumption
2.	$I \models (P \rightarrow Q) \land (Q \rightarrow R)$	by 1 and semantics of $\rightarrow$
3.	$I \not\models P \rightarrow R$	by 1 and semantics of $\rightarrow$
4.	$I \models P$	by 3 and semantics of $\rightarrow$
5.	$I \not\models R$	by 3 and semantics of $\rightarrow$
6.	$I \models P \rightarrow Q$	2 and semantics of $\land$
7.	$I \models Q \rightarrow R$	2 and semantics of $\land$
8.	$I \models Q$	by 4, 6, and modus ponens
9.	$I \models R$	by 8, 7, and modus ponens
10.	$I \models \bot$	5 and 9 are contradictory

Equivalence and Implication

- Two formulas F_1 and F_2 are equivalent

$$F_1 \iff F_2$$

  iff $F_1 \leftrightarrow F_2$ is valid, i.e., for all interpretations I, $I \models F_1 \leftrightarrow F_2$.
- Formula F_1 implies formula F_2

$$F_1 \implies F_2$$

  iff $F_1 \rightarrow F_2$ is valid, i.e., for all interpretations I, $I \models F_1 \rightarrow F_2$.
- $F_1 \iff F_2$ and $F_1 \implies F_2$ are not formulas. They are semantic assertions.
- We can check equivalence and implication by checking satisfiability.

Exercise

Prove that

$$\boldsymbol{R \land (\neg R \land P) \implies P}$$

Substitution

- A substitution σ is a mapping from formulas to formulas:

$$\sigma : \{F_1 \mapsto G_2, \ldots, F_n \mapsto G_n\}$$

- The domain of σ, $\mathbf{dom}(\sigma)$, is

$$\mathbf{dom}(\sigma) : \{F_1, \ldots, F_n\}$$

while the range $\mathbf{range}(\sigma)$ is

$$\mathbf{range}(\sigma) : \{G_1, \ldots, G_n\}$$

- The application of a substitution σ to a formula F, $F\sigma$, replaces each occurence of F_i with G_i. Replacements occur all at once.
- When two subformulas F_j and F_k are in $\mathbf{dom}(\sigma)$ and F_k is a strict subformula of F_j, then F_j is replaced first.

Example

Consider formula

$$F : P \wedge Q \rightarrow P \vee \neg Q$$

and substitution

$$\sigma : \{P \mapsto R, P \wedge Q \mapsto P \rightarrow Q\}$$

Then,

$$F\sigma : (P \rightarrow Q) \rightarrow R \vee \neg Q$$

Substitution

- A variable substitution is a substitution in which the domain consists only of propositional variables.
- When we write $F[F_1, \ldots, F_n]$, we mean that formula F can have formulas $F_1, \ldots, F_n$ as subformulas.
- If σ is $\{F_1 \mapsto G_1, \ldots, F_n \mapsto G_n\}$, then

$$F[F_1, \ldots, F_n]\sigma : F[G_1, \ldots, G_n]$$

- For example, in the previous example, writing

$$F[P, P \wedge Q]\sigma : F[R, P \to Q]$$

  emphasizes that P and $P \wedge Q$ are replaced by R and $P \to Q$, respectively.

Semantic Consequences of Substitution

Lemma (Substitution of Equivalent Formulas)

Consider substitution $\sigma : \{F_1 \mapsto G_1, \ldots, F_n \mapsto G_n\}$ *such that for each* i, $F_i \iff G_i$. *Then,* $F \iff F\sigma$.

Lemma (Valid Template)

If F *is valid and* $G = F\sigma$ *for some variable substitution* σ, *then* G *is valid.*

For example, because

$$F : (P \to Q) \leftrightarrow (\neg P \lor Q)$$

is valid, every formula of the form $F_1 \to F_2$ is equivalent to $\neg F_1 \lor F_2$, for arbitrary formulas F_1 and F_2.

Composition of Substitutions

Given substitutions σ_1 and σ_2, their composition $\sigma = \sigma_1\sigma_2$ ("apply σ_1 and then σ_2") is computed as follows:

1. Apply σ_2 to each formula of the range of σ_1, and add the results to σ.
2. If F_i of $F_i \mapsto G_i$ appears in the domain of σ_2 but not in the domain of σ_1, then add $F_i \mapsto G_i$ to σ.

For example,

$$\begin{aligned}
\sigma_1\sigma_2 : &\{P \mapsto R, P \wedge Q \mapsto P \to Q\}\{P \mapsto S, S \mapsto Q\} \\
&= \{P \mapsto R\sigma_2, P \wedge Q \mapsto (P \to Q)\sigma_2, S \mapsto Q\} \\
&= \{P \mapsto R, P \wedge Q \mapsto S \to Q, S \mapsto Q\}
\end{aligned}$$

Normal Forms

A normal form of formulas is a syntactic restriction such that for every formula of the logic, there is an equivalent formula in the normal form.

Three useful normal forms in logic:

- **Negation Normal Form (NNF)**
- **Disjunctive Normal Form (DNF)**
- **Conjunctive Normal Form (CNF)**

Negation Normal Form (NNF)

- NNF requires that $\neg$, $\wedge$, and $\vee$ are the only connectives (i.e., no $\rightarrow$ and $\leftrightarrow$) and that negations appear only in literals.
  - $P \wedge Q \wedge (R \vee \neg S)$
  - $\neg P \vee \neg(P \wedge Q))$
  - $\neg\neg P \wedge Q$
- Transforming a formula F to equivalent formula F' in NNF can be done by repeatedly applying the following list of template equivalences:

$$
\begin{array}{rcl}
\neg\neg F_1 & \iff & F_1 \\
\neg\top & \iff & \bot \\
\neg\bot & \iff & \top \\
\neg(F_1 \wedge F_2) & \iff & \neg F_1 \vee \neg F_2 \\
\neg(F_1 \vee F_2) & \iff & \neg F_1 \wedge \neg F_2 \\
F_1 \rightarrow F_2 & \iff & \neg F_1 \vee F_2 \\
F_1 \leftrightarrow F_2 & \iff & (F_1 \rightarrow F_2) \wedge (F_2 \rightarrow F_1)
\end{array}
$$

Exercise

Convert $F : \neg(P \rightarrow \neg(P \land Q))$ into NNF.

Disjunctive Normal Form (DNF)

- A formula is in disjunctive normal form (DNF) if it is a disjunction of conjunctive clauses (conjunctions of literals):

$$\bigvee_i \bigwedge_j l_{i,j}$$

- To convert a formula F into an equivalent formula in DNF, transform F into NNF and then distribute conjunctions over disjunctions:

$$\begin{array}{lcl} (F_1 \lor F_2) \land F_3 & \iff & (F_1 \land F_3) \lor (F_2 \land F_3) \\ F_1 \land (F_2 \lor F_3) & \iff & (F_1 \land F_2) \lor (F_1 \land F_3) \end{array}$$

Exercise

To convert

$$F : (Q_1 \vee \neg\neg Q_2) \wedge (\neg R_1 \rightarrow R_2)$$

into DNF,

- first transform it into NNF:
- then apply distributivity:

Conjunctive Normal Form (CNF)

- A formula is in conjunctive normal form (CNF) if it is a conjunction of clauses (i.e. conjunctions of disjunctions of literals):

$$\bigwedge_i \bigvee_j l_{i,j}$$

- To convert a formula F into an equivalent formula in DNF, transform F into NNF and distribute disjunctions over conjunctions:

$$\begin{array}{rcl} (F_1 \wedge F_2) \vee F_3 & \iff & (F_1 \vee F_3) \wedge (F_2 \vee F_3) \\ F_1 \vee (F_2 \wedge F_3) & \iff & (F_1 \vee F_2) \wedge (F_1 \vee F_3) \end{array}$$

- Exercise) Convert $F : (Q_1 \wedge \neg\neg Q_2) \vee (\neg R_1 \rightarrow R_2)$ into CNF.

Decision Procedures

- A **decision procedure** decides whether F is satisfiable after some finite steps of computation.
- Approaches for deciding satisfiability:
  - **Search**: exhaustively search through all possible assignments
  - **Deduction**: deduce facts from known facts by iteratively applying proof rules
  - **Combination**: Modern SAT solvers are based on DPLL that combines search and deduction in an effective way

Exhaustive Search

- The recursive algorithm for deciding satisfiability:

let rec **SAT** F =
  if $F = \top$ then true
  else if $F = \bot$ then false
  else
    let P = **Choose**(**vars**(F)) in
    (**SAT** $F\{P \mapsto \top\}$) $\vee$ (**SAT** $F\{P \mapsto \bot\}$)

- When applying $F\{P \mapsto \top\}$ and $F\{P \mapsto \bot\}$, the resulting formulas should be simplified using template equivalences on PL.

$$\begin{array}{ccc} \top \iff \neg\bot & \bot \iff \neg\top & \neg\neg F \iff F \\ F \wedge \top \iff F & F \wedge \bot \iff \bot & F \wedge F \iff F \\ F \vee \top \iff \top & F \vee \bot \iff F & F \vee F \iff F \end{array}$$

$$\cdots$$

Example

$$F : (P \to Q) \land P \land \neg Q$$

- Choose variable P and

$$F\{P \mapsto \top\} : (\top \to Q) \land \top \land \neg Q$$

which simplifies to

$$F_1 : Q \land \neg Q$$

  - $F_1\{Q \mapsto \top\} : \bot$
  - $F_1\{Q \mapsto \bot\} : \bot$

- Recurse on the other branch for P in F:

$$F\{P \mapsto \bot\} : (\bot \to Q) \land \bot \land \neg Q$$

which simplifies to $\bot$.

- All branches end without finding a satisfying assignment.

Example

$$F : (P \to Q) \land \neg P$$

- Choose P and recurse on the first case:

$$F\{P \mapsto \top\} : (\top \to Q) \land \neg T$$

  which is equivalent to $\bot$.
- Try the other case:

$$F\{P \to \bot\} : (\bot \to Q) \land \neg\bot$$

  which is equivalent to $\top$.
- Arbitrarily assigning a value to Q produces the satisfying interpretation:

$$I : \{P \mapsto \textbf{false}, Q \mapsto \textbf{true}\}.$$

Equisatisfiability

- SAT solvers convert a given formula F to CNF.
- Conversion to an equivalent CNF incurs exponential blow-up in worst-case.
- F is converted to an equisatisfiable CNF formula, which increases the size by only a constant factor.
- F and F' are **equisatisfiable** when F is satisfiable iff F' is satisfiable.
- Equisatisfiability is a weaker notion of equivalence, which is still useful when deciding satisfiability.

Conversion to an Equisatisfiable Formula in CNF

- Idea: Introduce new variables to represent the subformulas of F with extra clauses that assert that these new variables are equivalent to the subformulas that they represent.
- $F : x_1 \to (x_2 \land x_3)$
  - Introduce two variables a_1 and a_2 with two equivalences:

$$\begin{array}{lcl} a_1 & \leftrightarrow & (x_1 \to a_2) \\ a_2 & \leftrightarrow & (x_2 \land x_3) \end{array}$$

    We need to satisfy a_1, together with the above two equivalences.
  - Convert the equivalences to CNF:

$$\begin{array}{c} (a_1 \lor x_1) \land (a_1 \lor \neg a_2) \land (\neg a_1 \lor \neg x_1 \lor a_2) \\ (\neg a_2 \lor x_2) \land (\neg a_2 \lor x_3) \land (a_2 \lor \neg x_2 \lor \neg x_3) \end{array}$$

  - The final CNF formula:

$$\begin{array}{c} F' = a_1 \land (a_1 \lor x_1) \land (a_1 \lor \neg a_2) \land (\neg a_1 \lor \neg x_1 \lor a_2) \land \\ (\neg a_2 \lor x_2) \land (\neg a_2 \lor x_3) \land (a_2 \lor \neg x_2 \lor \neg x_3) \end{array}$$

  - F is satisfiable iff F' is satisfiable

The Resolution Procedure

- Applicable only to CNF formulas.
- Observation: to satisfy clauses $C_1[P]$ and $C_2[\neg P]$ that share variable P but disagree on its value, either the rest of C_1 or the rest of C_2 must be satisfied. Why?
- The clause $C_1[\bot] \vee C_2[\bot]$ (with simplification) can be added as a conjunction to F to produce an equivalent formula still in CNF.
- The proof rule for **clausal resolution**:

$$\frac{C_1[P] \quad C_2[\neg P]}{C_1[\bot] \vee C_2[\bot]}$$

  The new clause $C_1[\bot] \vee C_2[\bot]$ is called the **resolvent**.
- If ever $\bot$ is deduced via resolution, F must be unsatisfiable. Otherwise, if no further resolutions are possible, F must be satisfiable.

Examples

$$F : (\neg P \lor Q) \land P \land \neg Q$$

- From resolution

$$\frac{(\neg P \lor Q) \qquad P}{Q},$$

construct $(\neg P \lor Q) \land P \land \neg Q \land Q$. From resolution

$$\frac{\neg Q \qquad Q}{\bot}$$

deduce that F is unsatisfiable.

Examples

$$F : (\neg P \vee Q) \wedge \neg Q)$$

- The resolution procedure yields

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

  No further resolutions are possible. F is satisfiable.
- A satisfying interpretation:

  $$I : \{P \mapsto \textbf{false}, Q \mapsto \textbf{false}\}$$

- A CNF formula that does not contain the clause $\perp$ and to which no more resolutions are applicable represents all possible satisfying interpretations.

DPLL

- The Davis-Putnam-Logemann-Loveland algorithm (DPLL) combines the enumerative search and a restricted form of resolution, called **unit resolution**:

$$\frac{l \qquad C[\neg l]}{C[\bot]}$$

  where l is a literal ($l = P$ or $l = \neg P$).
- The process of applying this resolution as much as possible is called **Boolean constraint propagation (BCP)**.

BCP Example

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

- Apply unit resolution

$$\frac{P \qquad (\neg P \vee Q)}{Q}$$

to produce $F' : Q \wedge (R \vee \neg Q \vee S)$. Applying unit resolution

$$\frac{Q \qquad R \vee \neg Q \vee S}{R \vee S}$$

produces $F'' : R \vee S$, ending this round of BCP.

DPLL

DPLL is similar to SAT, except that it begins by applying BCP:

let rec **DPLL** F =
  let F' = **BCP**(F) in
  if $F' = \top$ then true
  else if $F' = \bot$ then false
  else
    let P = **Choose**(**vars**(F')) in
    (**DPLL** $F'\{P \mapsto \top\}$) $\vee$ (**DPLL** $F'\{P \mapsto \bot\}$)

Pure Literal Propagation (PLP)

- If variable P appears only positively or only negatively in F, remove all clauses containing an instance of P.
  - If P appears only positively (i.e. no $\neg P$ in F), replace P by $\top$.
  - If P appears only negatively (i.e. no P in F), replace P by $\bot$.
- The resulting formula F' is equisatisfiable to F.
- When only such pure variables remain, the formula must be satisfiable. A full interpretation can be constructed by setting each variable's value based on whether it appears only positively (true) or only negatively (false).

Example) $F : (\neg P \vee Q) \wedge (R \vee \neg Q \vee S)$.

- P appears only negatively in F

$$F' : (R \vee \neg Q \vee S)$$

- R and S appear only positively in F

$$F' : (\neg P \vee Q)$$

DPLL with PLP

let rec **DPLL** F =
  let F' = **PLP**(**BCP**(F)) in
  if $F' = \top$ then true
  else if $F' = \bot$ then false
  else
    let P = **Choose**(**vars**(F')) in
    (**DPLL** $F'\{P \mapsto \top\}$) $\vee$ (**DPLL** $F'\{P \mapsto \bot\}$)

Example 1

$$F : P \land (\neg P \lor Q) \land (R \lor \neg Q \lor S)$$

1. Applying BCP produces
$$F'' : R \lor S$$
2. All variables occur positively, so F is satisfiable.
3. A satisfying interpretation:

$$\{P \mapsto \textbf{true}, Q \mapsto \textbf{true}, R \mapsto \textbf{true}, S \mapsto \textbf{true}\}$$

Example 2

$$F : (\neg P \lor Q \lor R) \land (\neg Q \lor R) \land (\neg Q \lor \neg R) \land (P \lor \neg Q \lor \neg R)$$

- No BCP and PLP are applicable.
- Choose Q to branch on:

$$F\{Q \mapsto \top\} : R \land (\neg R) \land (P \lor \neg R)$$

The unit resolution with R and $\neg R$ deduces $\bot$, finishing this branch.

- On the other branch for Q:

$$F\{Q \mapsto \bot\} : (\neg P \lor R)$$

P and R are pure, so the formula is satisfiable. A satisfying interpretation:

$$I : \{P \mapsto \textbf{false}, Q \mapsto \textbf{false}, R \mapsto \textbf{true}\}$$

Summary

- Syntax and semantics of propositional logic
- Satisfiability and validity
- Equivalence, implications, and equisatisfiability
- Substitution
- Normal forms: NNF, DNF, CNF
- Decision procedures for satisfiability