Introduction to SAT, Predicate Logic and DPLL solving

Lecture 1

Nikolaj Bjørner
Microsoft Research
DTU Winter course January 2\textsuperscript{nd} 2012
Organized by Flemming Nielson & Hanne Riis Nielson
General overview of propositional and predicate logic

Refresher on SAT and modern DPLL
The syntax and semantics of propositional and predicate logic.

Algorithmic principles of modern SAT solvers:
- DPLL (actually DLL) algorithm
- Conflict Directed Clause Learning (CDCL)
- Two-watch literal indexing
Logic is the art and science of effective reasoning.

How can we draw general and reliable conclusions from a collection of facts?

**Formal logic**: Precise, syntactic characterizations of well-formed expressions and valid deductions.

Formal logic makes it possible to calculate consequences at the symbolic level.

Computers can be used to automate such symbolic calculations.
Logic studies the relationship between language, meaning, and (proof) method.

A logic consists of a language in which (well-formed) sentences are expressed.

A semantic that distinguishes the valid sentences from the refutable ones.

A proof system for constructing arguments justifying valid sentences.

Examples of logics include propositional logic, equational logic, first-order logic, higher-order logic, and modal logics.
What is logical language?

- A language consists of logical symbols whose interpretations are fixed, and non-logical ones whose interpretations vary.
- These symbols are combined together to form well-formed formulas.
- In propositional logic PL, the connectives $\land$, $\lor$, and $\neg$ have a fixed interpretation, whereas the constants $p$, $q$, $r$ may be interpreted at will.
SMT: Basic Architecture

SAT + Theory Solvers = SMT

- Equality + UF
- Arithmetic
- Bit-vectors
- ...

Case Analysis
Propositional Logic

Formulas: \( \varphi := p \mid \varphi_1 \lor \varphi_2 \mid \varphi_1 \land \varphi_2 \mid \neg \varphi_1 \mid \varphi_1 \Rightarrow \varphi_2 \)

Examples:
\[ p \lor q \Rightarrow q \lor p \]
\[ p \land \neg q \land (\neg p \lor q) \]

We say \( p \) and \( q \) are propositional variables.
An interpretation $M$ assigns values \{true, false\} to propositional variables.

Let $F$ and $G$ range over $PL$ formulas.

<table>
<thead>
<tr>
<th>$F$</th>
<th>$G$</th>
<th>$F \lor G$</th>
<th>$F \land G$</th>
<th>$F \Rightarrow G$</th>
<th>$\neg F$</th>
</tr>
</thead>
<tbody>
<tr>
<td>false</td>
<td>false</td>
<td>false</td>
<td>false</td>
<td>true</td>
<td>true</td>
</tr>
<tr>
<td>false</td>
<td>true</td>
<td>true</td>
<td>false</td>
<td>true</td>
<td>true</td>
</tr>
<tr>
<td>true</td>
<td>false</td>
<td>true</td>
<td>false</td>
<td>false</td>
<td>false</td>
</tr>
<tr>
<td>true</td>
<td>true</td>
<td>true</td>
<td>true</td>
<td>true</td>
<td>false</td>
</tr>
</tbody>
</table>
Satisfiability & Validity

- A formula is **satisfiable** if it has an interpretation that makes it logically true.
- In this case, we say the interpretation is a **model**.
- A formula is **unsatisfiable** if it does not have any model.
- A formula is **valid** if it is logically true in any interpretation.
- A propositional formula is valid if and only if its negation is unsatisfiable.
Satisfiability & Validity: examples

$p \lor q \Rightarrow q \lor p$

$p \lor q \Rightarrow q$

$p \land \neg q \land (\neg p \lor q)$

<table>
<thead>
<tr>
<th></th>
<th></th>
<th>$F \lor G$</th>
<th>$F \land G$</th>
<th>$F \Rightarrow G$</th>
<th>$\neg F$</th>
</tr>
</thead>
<tbody>
<tr>
<td>false</td>
<td>false</td>
<td>false</td>
<td>false</td>
<td>true</td>
<td>true</td>
</tr>
<tr>
<td>false</td>
<td>true</td>
<td>true</td>
<td>false</td>
<td>true</td>
<td>true</td>
</tr>
<tr>
<td>true</td>
<td>false</td>
<td>true</td>
<td>false</td>
<td>false</td>
<td>false</td>
</tr>
<tr>
<td>true</td>
<td>true</td>
<td>true</td>
<td>true</td>
<td>true</td>
<td>false</td>
</tr>
</tbody>
</table>
Satisfiability & Validity: examples

\( p \lor q \Rightarrow q \lor p \)  
\text{VALID}

\( p \lor q \Rightarrow q \)  
\text{SATISFIABLE}

\( p \land \neg q \land (\neg p \lor q) \)  
\text{UNSATISFIABLE}

<p>| | | | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>\text{false}</td>
<td>\text{false}</td>
<td>\text{false}</td>
<td>\text{false}</td>
<td>\text{true}</td>
<td>\text{true}</td>
</tr>
<tr>
<td>\text{false}</td>
<td>\text{true}</td>
<td>\text{true}</td>
<td>\text{false}</td>
<td>\text{true}</td>
<td>\text{true}</td>
</tr>
<tr>
<td>\text{true}</td>
<td>\text{false}</td>
<td>\text{true}</td>
<td>\text{false}</td>
<td>\text{false}</td>
<td>\text{false}</td>
</tr>
<tr>
<td>\text{true}</td>
<td>\text{true}</td>
<td>\text{true}</td>
<td>\text{true}</td>
<td>\text{true}</td>
<td>\text{false}</td>
</tr>
</tbody>
</table>
We say two formulas $F$ and $G$ are *equivalent* if and only if they evaluate to the same value (true or false) in every interpretation.

\[
\neg (F \land G) \text{ is equivalent to } \neg F \lor \neg G
\]
\[
\neg (F \lor G) \text{ is equivalent to } \neg F \land \neg G
\]
\[
\neg F \Rightarrow G \text{ is equivalent to } \neg G \Rightarrow F
\]
\[
F \lor (G \land H) \text{ is equivalent to } (F \lor G) \land (F \lor H)
\]
\[
F \land (G \lor H) \text{ is equivalent to } (F \land G) \lor (F \land H)
\]
Equisatisfiable

We say formulas $A$ and $B$ are **equisatisfiable** if and only if $A$ is satisfiable if and only if $B$ is.

During this tutorial, we will describe transformations that preserve equivalence and equisatisfiability.
Normal Forms

A formula where negation is applied only to propositional atoms is said to be in negation normal form (NNF).

A literal is either a propositional atom or its negation.

A formula that is a multiary conjunction of multiary disjunctions of literals is in conjunctive normal form (CNF).

A formula that is a multiary disjunction of multiary conjunctions of literals is in disjunctive normal form (DNF).

Exercise 3  Show that every propositional formula is equivalent to one in NNF, CNF, and DNF.

Exercise 4  Show that every $n$-ary Boolean function can be expressed using just $\neg$ and $\vee$. 
NNF?

\((p \lor \neg q) \land (q \lor \neg (r \land \neg p))\)
NNF? NO

\((p \lor \neg q) \land (q \lor \neg (r \land \neg p))\)
Normal Forms

NNF? NO

$$(p \lor \lnot q) \land (q \lor \lnot (r \land \lnot p))$$

1. $\lnot \lnot A \iff A$

2. $A \Rightarrow B \iff \lnot A \lor B$

3. $\lnot (A \land B) \iff \lnot A \lor \lnot B$

4. $\lnot (A \lor B) \iff \lnot A \land \lnot B$
Normal Forms

NNF? NO

$$(p \lor \neg q) \land (q \lor \neg (r \land \neg p))$$

$$\iff$$

$$(p \lor \neg q) \land (q \lor (\neg r \lor \neg \neg p))$$

$$1. \quad \neg
\neg A \iff A$$

$$2. \quad A \Rightarrow B \iff \neg A \lor B$$

$$3. \quad \neg (A \land B) \iff \neg A \lor \neg B$$

$$4. \quad \neg (A \lor B) \iff \neg A \land \neg B$$
Normal Forms

NNF? NO

\[(p \lor \neg q) \land (q \lor \neg (r \land \neg p))\]

\[\iff\]

\[(p \lor \neg q) \land (q \lor (\neg r \lor \neg \neg p))\]

\[\iff\]

\[(p \lor \neg q) \land (q \lor (\neg r \lor p))\]

1. \(\neg\neg A \iff A\)

2. \(A \Rightarrow B \iff \neg A \lor B\)

3. \(\neg(A \land B) \iff \neg A \lor \neg B\)

4. \(\neg(A \lor B) \iff \neg A \land \neg B\)
Normal Forms

CNF?

\((p \land s) \lor (\neg q \land r) \land (q \lor \neg p \lor s) \land (\neg r \lor s)\)
Normal Forms

CNF? NO

\(((p \land s) \lor (\neg q \land r)) \land (q \lor \neg p \lor s) \land (\neg r \lor s)\)
CNF? NO

\[(p \land s) \lor (\neg q \land r) \land (q \lor \neg p \lor s) \land (\neg r \lor s)\]

Distributivity

1. \[A \lor (B \land C) \iff (A \lor B) \land (A \lor C)\]
2. \[A \land (B \lor C) \iff (A \land B) \lor (A \land C)\]
Normal Forms

CNF? NO

\[(p \land s) \lor (\neg q \land r) \land (q \lor \neg p \lor s) \land (\neg r \lor s)\]

\[
\iff
\]

\[(p \land s) \lor (\neg q) \land ((p \land s) \lor r) \land (q \lor \neg p \lor s) \land (\neg r \lor s)\]

Distributivity

1. \(A \lor (B \land C) \iff (A \lor B) \land (A \lor C)\)
2. \(A \land (B \lor C) \iff (A \land B) \lor (A \land C)\)
Normal Forms

CNF? NO

\[(p \land s) \lor (\neg q \land r) \land (q \lor \neg p \lor s) \land (\neg r \lor s)\]

\[\iff\]

\[((p \land s) \lor \neg q) \land ((p \land s) \lor r) \land (q \lor \neg p \lor s) \land (\neg r \lor s)\]

\[\iff\]

\[(p \lor \neg q) \land (s \lor \neg q) \land ((p \land s) \lor r) \land (q \lor \neg p \lor s) \land (\neg r \lor s)\]

Distributivity

1. \(A \lor (B \land C) \iff (A \lor B) \land (A \lor C)\)
2. \(A \land (B \lor C) \iff (A \land B) \lor (A \land C)\)
Normal Forms

CNF? NO

\[(p \land s) \lor (\neg q \land r) \land (q \lor \neg p \lor s) \land (\neg r \lor s)\]

\[\equiv\]

\[((p \land s) \lor \neg q) \land ((p \land s) \lor r) \land (q \lor \neg p \lor s) \land (\neg r \lor s)\]

\[\equiv\]

\[(p \lor \neg q) \land (s \lor \neg q) \land ((p \land s) \lor r) \land (q \lor \neg p \lor s) \land (\neg r \lor s)\]

\[\equiv\]

\[(p \lor \neg q) \land (s \lor \neg q) \land (p \lor r) \land (s \lor r) \land (q \lor \neg p \lor s) \land (\neg r \lor s)\]
DNF?

\[ p \land (\neg p \lor q) \land (\neg q \lor r) \]
Normal Forms

DNF? NO, actually this formula is in CNF

\[ p \land (\neg p \lor q) \land (\neg q \lor r) \]
DNF? NO, actually this formula is in CNF

\[ p \land (\neg p \lor q) \land (\neg q \lor r) \]

Distributivity

1. \[ A \lor (B \land C) \iff (A \lor B) \land (A \lor C) \]
2. \[ A \land (B \lor C) \iff (A \land B) \lor (A \land C) \]
DNF? NO, actually this formula is in CNF

\[ p \land (\neg p \lor q) \land (\neg q \lor r) \]

\[ \iff \]

\[ ((p \land \neg p) \lor (p \lor q)) \land (\neg q \lor r) \]

**Distributivity**

1. \( A \lor (B \land C) \iff (A \lor B) \land (A \lor C) \)
2. \( A \land (B \lor C) \iff (A \land B) \lor (A \land C) \)
DNF? NO, actually this formula is in CNF

\[ p \land (\neg p \lor q) \land (\neg q \lor r) \]

\[ \iff \]

\[ ( (p \land \neg p) \lor (p \lor q) ) \land (\neg q \lor r) \]

\[ \iff \]

\[ (p \lor q) \land (\neg q \lor r) \]

Distributivity
1. \[ A \lor (B \land C) \iff (A \lor B) \land (A \lor C) \]
2. \[ A \land (B \lor C) \iff (A \land B) \lor (A \land C) \]

Other Rules
1. \[ A \land \neg A \iff \bot \]
2. \[ A \lor \bot \iff A \]
Normal Forms

DNF? NO, actually this formula is in CNF

\[ p \land (\neg p \lor q) \land (\neg q \lor r) \]
\[ \iff \]
\[ ((p \land \neg p) \lor (p \lor q)) \land (\neg q \lor r) \]
\[ \iff \]
\[ (p \lor q) \land (\neg q \lor r) \]
\[ \iff \]
\[ ((p \lor q) \land \neg q) \lor ((p \lor q) \land r) \]

Distributivity

1. \( A \lor (B \land C) \iff (A \lor B) \land (A \lor C) \)
2. \( A \land (B \lor C) \iff (A \land B) \lor (A \land C) \)

Other Rules

1. \( A \land \neg A \iff \bot \)
2. \( A \lor \bot \iff A \)
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:

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

\[
\frac{x \Leftrightarrow l_i \lor l_j}{\neg x \lor l_i \lor l_j, \neg l_i \lor x, \neg l_j \lor x}
\]

\[
\frac{x \Leftrightarrow l_i \land l_j}{\neg x \lor l_i, \neg x \lor l_j, \neg l_i \lor \neg l_j \lor x}
\]

(*) \( x \) must be a fresh variable.

Ex: Show that the rules preserve equisatisfiability.
Translation of \((p \land (q \lor r)) \lor t\):

\[
\begin{align*}
(p \land (q \lor r)) \lor t \\
(p \land x_1) \lor t, x_1 \iff q \lor r \\
x_2 \lor t, x_2 \iff p \land x_1, x_1 \iff q \lor r \\
x_2 \lor t, \neg x_2 \lor p, \neg x_2 \lor x_1, \neg p \lor \neg x_1 \lor x_2, x_1 \iff q \lor r \\
x_2 \lor t, \neg x_2 \lor p, \neg x_2 \lor x_1, \neg p \lor \neg x_1 \lor x_2, \neg x_1 \lor q \lor r, \neg q \lor x_1, \neg r \lor x_1
\end{align*}
\]

Ex: Implement a CNF translator.
Resolution

Formula must be in CNF.

Resolution procedure uses only one rule:

\[
\begin{align*}
C_1 \lor p, C_2 \lor \neg p \\
\hline
C_1 \lor p, C_2 \lor \neg p, C_1 \lor C_2
\end{align*}
\]

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 \( p \lor q \lor r \), and \( \neg p \lor r \lor t \) is \( q \lor r \lor t \).

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

Ex: Show that the resolution rule is sound.
A refutation of \( \neg p \vee \neg q \vee r, \ p \vee r, \ q \vee r, \ \neg r \):

Ex: Implement a naïve resolution procedure.
Unit & Input Resolution

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

\[
\frac{C \lor \bar{t}, l}{C, l} \text{unit}
\]

Unit resolution always *decreases* the configuration *size* (\(C \lor \bar{t}\) 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).
DPLL = Unit resolution + Split rule.

\[
\frac{\Gamma}{\Gamma, p | \Gamma, \neg p} \quad \text{split} \quad \text{p and } \neg p \text{ are not in } \Gamma.
\]

\[
\frac{C \lor \bar{l}, l}{C, l} \quad \text{unit}
\]

Used in the most efficient SAT solvers.
Pure Literals

A literal is **pure** if it only occurs positively or negatively.

Example:
\[
\varphi = (\neg x_1 \lor x_2) \land (x_3 \lor \neg x_2) \land (x_4 \lor \neg x_5) \land (x_5 \lor \neg x_4)
\]

\(\neg x_1\) and \(x_3\) are pure literals

**Pure literal rule:**
Clauses containing pure literals can be removed from the formula (i.e. just satisfy those pure literals)

\[
\varphi_{\neg x_1,x_3} = (x_4 \lor \neg x_5) \land (x_5 \lor \neg x_4)
\]

Preserve satisfiability, not logical equivalency!
A literal is **pure** if only occurs positively or negatively.

Example:

$$\varphi = (\neg x_1 \lor x_2) \land (x_3 \lor \neg x_2) \land (x_4 \lor \neg x_5) \land (x_5 \lor \neg x_4)$$

$\neg x_1$ and $x_3$ are pure literals

**Pure literal rule:**

Clauses containing pure literals can be removed from the formula (i.e. just satisfy those pure literals)

$$\varphi_{\neg x_1, x_3} = (x_4 \lor \neg x_5) \land (x_5 \lor \neg x_4)$$

Preserve satisfiability, not logical equivalency!
DPLL (as a procedure)

- Standard **backtrack search**
- **DPLL(F):**
  - Apply unit propagation
  - If conflict identified, return **UNSAT**
  - Apply the pure literal rule
  - If F is satisfied (empty), return **SAT**
  - Select decision variable \( x \)
    - If \( \text{DPLL}(F \land x) = \text{SAT} \) return **SAT**
    - return \( \text{DPLL}(F \land \neg x) \)
DPLL

M | F

Partial model

Set of clauses
Guessing

\[ p | p \lor q, \neg q \lor r \]

\[ p, \neg q | p \lor q, \neg q \lor r \]
Deducing

\[ p \mid p \lor q, \neg p \lor s \]

\[ p, s \mid p \lor q, \neg p \lor s \]
Backtracking

\[
p, \neg s, \ q \ \mid \ p \lor q, \ s \lor q, \ \neg p \lor \neg q
\]

\[
p, \ s \ \mid \ p \lor q, \ s \lor q, \ \neg p \lor \neg q
\]
Modern DPLL

- Efficient indexing (two-watch literal)
- Non-chronological backtracking (backjumping)
- Lemma learning
### Modern DPLL in a nutshell

<table>
<thead>
<tr>
<th>Operation</th>
<th>Rule</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Initialize</strong></td>
<td>$\epsilon \mid F$</td>
<td>$F$ is a set of clauses</td>
</tr>
<tr>
<td><strong>Decide</strong></td>
<td>$M \mid F \Rightarrow M, \ell \mid F$</td>
<td>$\ell$ is unassigned</td>
</tr>
<tr>
<td><strong>Propagate</strong></td>
<td>$M \mid F, C \lor \ell \Rightarrow M, \ell^{C\lor\ell} \mid F, C \lor \ell$</td>
<td>$C$ is false under $M$</td>
</tr>
<tr>
<td><strong>Conflict</strong></td>
<td>$M \mid F, C \Rightarrow M \mid F, C \mid C$</td>
<td>$C$ is false under $M$</td>
</tr>
<tr>
<td><strong>Resolve</strong></td>
<td>$M \mid F \mid C' \lor \neg \ell \Rightarrow M \mid F \mid C' \lor C$</td>
<td>$\ell^{C\lor\ell} \in M$</td>
</tr>
<tr>
<td><strong>Learn</strong></td>
<td>$M \mid F \mid C \Rightarrow M \mid F, C \mid C$</td>
<td></td>
</tr>
<tr>
<td><strong>Backjump</strong></td>
<td>$M \neg \ell M' \mid F \mid C \lor \ell \Rightarrow M \ell^{C\lor\ell} \mid F$</td>
<td>$C$ has no literals in $M'$</td>
</tr>
<tr>
<td><strong>Unsat</strong></td>
<td>$M \mid F \mid \emptyset \Rightarrow Unsat$</td>
<td></td>
</tr>
<tr>
<td><strong>Sat</strong></td>
<td>$M \mid F \Rightarrow M$</td>
<td>$F$ true under $M$</td>
</tr>
<tr>
<td><strong>Restart</strong></td>
<td>$M \mid F \Rightarrow \epsilon \mid F$</td>
<td></td>
</tr>
</tbody>
</table>

Adapted and modified from [Nieuwenhuis, Oliveras, Tinelli J.ACM 06]
Lemma learning

\[ \neg t, p, q, s \mid t \lor \neg p \lor q, \neg q \lor s, \neg p \lor \neg s \]

\[ \neg t, p, q, s \mid t \lor \neg p \lor q, \neg q \lor s, \neg p \lor \neg s \mid \neg p \lor \neg s \]

\[ \neg t, p, q, s \mid t \lor \neg p \lor q, \neg q \lor s, \neg p \lor \neg s \mid \neg p \lor \neg q \]

\[ \neg t, p, q, s \mid t \lor \neg p \lor q, \neg q \lor s, \neg p \lor \neg s \mid \neg p \lor t \]
Two-watch literals

- Indexing is key to efficient theorem proving.
  - An index is a dictionary that is tuned to search.
- The literal two-watch scheme is the main index in modern SAT solvers.
  - Track two literals per clause, such that:
    - Track literals that are unassigned or assigned to true.
    - If one of the tracked literals is assigned to false, search for non-tracked literal that is either unassigned or true.
    - If there is no other unassigned or true literal to select, then the other watched literal can be assigned to true.
  - Unit propagation or conflict detection.
Watch literals for Propagate and Conflict

Naïve: For every literal $l$ maintain map:

$\text{Watch}(l) = \{C_1 \ldots C_m\}$ where $\neg l \in C_i$

If $l$ is assigned to true, check each $C_j \in \text{Watch}(l)$ for Conflict or Propagate

But most of the time, some other literal in $C_j$ is either:

- Unassigned (not yet assigned)
- Assigned to true.
Insight:

- No need to include clause $C$ in every set $\text{Watch}(l)$ where $\neg l \in C$.

- It suffices to include $C$ in at most 2 such sets.

Maintain invariant:

- If some literal $l$ in $C$ unassigned, or assigned to $true$, then $C$ belongs to the $\text{Watch}(l')$ of some literal that is unassigned or true.
Modern DPLL - implementation
Maintain 2-watch invariant:

- Set \( l \) to true (\(-l\) to false).
- For each \( C \in \text{Watch}(l) \)
  - If all literals in \( C \) are assigned to false, then **Backjump**
  - Else, if all but one literal in \( C \) is assigned to false, then **Propagate**
  - Else, if the other literal in \( l' \in C \) where \( C \in \text{Watch}(l') \) is assigned to true, then do nothing.
  - Else, some other literal \( l' \) is true or unassigned, and not watched. Set \( \text{Watch}(l') \leftarrow \text{Watch}(l') \cup \{ C \} \), set \( \text{Watch}(l) \leftarrow \text{Watch}(l) \setminus \{ C \} \).
Heuristic: Phase caching

- Remember the last truth value assigned to propositional atom.
- If using rule **Decide**, then re-use the old assignment.

Why should this be good (in practice)?

- Dependencies follow *clusters*.
- Truth values in a cluster are dependent.
- Truth values between clusters are independent.
- **Decide** is mainly used when jumping between clusters.
Modern DPLL - tuning

- Tune between different heuristics:
  - Restart frequency
    - Why is restarting good?
  - Phase to assign to decision variable
  - Which variable to split on
    - Use simulated annealing based on activity in conflicts
    - Feedback factor from phase changes
  - Which lemmas to learn
    - Not necessarily unique
    - Minimize lemmas
    - Sub-sumption
  - Blocked clause elimination
  - Cache binary propagations
Formulas and Queries

Find satisfying instance.
- Incomplete methods (Stochastic local search)

Establish that there are no satisfying instance.
- Complete methods

Simplify formula:
- Normal form conversion: NNF, CNF, DNF, BDD, d-DNF,..
- Quantifier elimination

Compute interpolants between two formulas.
\[ A \Rightarrow I \Rightarrow B, L(I) \subseteq L(A) \cap L(B). \]

Compute fixed-points of predicate transformers.