Efficient Transformation Into CNF

SAT solvers assume that the WFF to be tested for satisfiability is in CNF.

Before applying one of the modern SAT solvers, we need therefore an efficient way of translating an arbitrary propositional WFF into an **equisatisfiable** (not necessarily **equivalent**) WFF in CNF.

If $\varphi$ is a propositional WFF in CNF, we may write

$$\varphi = \{C_1, \ldots, C_n\}, \quad \text{i.e., a finite set of clauses}$$

instead of $\varphi = C_1 \land \cdots \land C_n$ where each $C_i$ is a disjunction of literals.
Efficient Transformation Into CNF

(A) BNF definition of arbitrary propositional WFF’s:

\[ \varphi ::= p \mid \neg \varphi \mid \varphi_1 \land \varphi_2 \mid \varphi_1 \lor \varphi_2 \]

(B) Another BNF definition of propositional WFF’s, sometimes said to be in negation normal form (NNF):

\[ \varphi ::= p \mid \neg p \mid \varphi_1 \land \varphi_2 \mid \varphi_1 \lor \varphi_2 \]

The difference between (A) and (B) is that (B) restricts where “\(\neg\)” may appear: In (B), every “\(\neg\)” immediately precedes a propositional atom \(p\).

Lemma

Every propositional WFF \(\varphi\) in the syntax of BNF definition (A) can be translated in linear time into an equivalent propositional WFF \(\psi\) in the syntax of BNF definition (B) such that \(|\psi| < (3/2) \cdot |\varphi|\).

Proof. Left to the reader.
Efficient Transformation Into CNF

The definition of $\text{CNF}(\ )$ is by induction on WFF’s. Because it is inductive, it translates into a recursive algorithm, where $\Delta$ is a finite set of clauses:\(^1\)

1. $\text{CNF}(p, \Delta) := \langle p, \Delta \rangle$

2. $\text{CNF}(\neg \varphi, \Delta) := \langle \neg \ell, \Delta' \rangle$ where $\text{CNF}(\varphi, \Delta) = \langle \ell, \Delta' \rangle$

3. $\text{CNF}(\varphi_1 \land \varphi_2, \Delta) := \langle p, \Delta' \rangle$ where
   
   \[\text{CNF}(\varphi_1, \Delta) = \langle \ell_1, \Delta_1 \rangle, \quad \text{CNF}(\varphi_2, \Delta_1) = \langle \ell_2, \Delta_2 \rangle,\]
   
   $p$ is a fresh atom (propositional variable),
   
   $\Delta' = \Delta_2 \cup \{\neg p \lor \ell_1, \neg p \lor \ell_2, \neg \ell_1 \lor \neg \ell_2 \lor p\}$

4. $\text{CNF}(\varphi_1 \lor \varphi_2, \Delta) := \langle p, \Delta' \rangle$ where
   
   \[\text{CNF}(\varphi_1, \Delta) = \langle \ell_1, \Delta_1 \rangle, \quad \text{CNF}(\varphi_2, \Delta_1) = \langle \ell_2, \Delta_2 \rangle,\]
   
   $p$ is a fresh atom (propositional variable),
   
   $\Delta' = \Delta_2 \cup \{\neg p \lor \ell_1 \lor \ell_2, \neg \ell_1 \lor p, \neg \ell_2 \lor p\}$

(If you prefer, every “$:=”$ above can be replaced by “return”.)

---

Efficient Transformation Into CNF

Theorem
Let $\varphi$ be an arbitrary propositional WFF and let $\text{CNF}(\varphi, \emptyset) = \langle \ell, \Delta \rangle$. Then $\varphi$ is satisfiable iff $\{\ell\} \cup \Delta$ is satisfiable.

Proof.
Left to the reader.

Exercise
Carry out the transformation $\text{CNF}(\varphi, \emptyset)$ where

$$\varphi := \neg \left( (q_1 \lor \neg q_2) \land q_3 \right)$$

Exercise
Search the Web for improvements on the transformation $\text{CNF}(\cdot)$. (How about introducing multi-arity $\land$ and multi-arity $\lor$?)
Two main approaches:

- **SAT solvers based on stochastic search**: The solver first guesses a full assignment (also called full valuation), i.e., an assignment of truth-values to all atoms. If the WFF evaluates to F under this assignment, it starts to flip truth-values of the atoms according to some heuristics. Typically, it counts the number of unsatisfied clauses and chooses the flip that minimizes this number.

---

2 Search the Web for articles on “resolution” and “paramodulation” (a variation on resolution), sometimes in articles on general “automated theorem proving”.
Approaches to SAT Solvers

Two main approaches:

- **SAT solvers based on **stochastic search**: The solver first guesses a full assignment (also called full valuation), i.e., an assignment of truth-values to all atoms. If the WFF evaluates to $F$ under this assignment, it starts to flip truth-values of the atoms according to some heuristics. Typically, it counts the number of unsatisfied clauses and chooses the flip that minimizes this number.

- **SAT solvers based on exhaustive search**: The solver traverses a binary tree, in which internal nodes are partial valuations and leaves are full valuations, and repeatedly backtracks in search of a satisfying full valuation. Several sub-approaches:

  2. Search the Web for articles on “resolution” and “paramodulation” (a variation on resolution), sometimes in articles on general “automated theorem proving”.

Approaches to SAT Solvers

Two main approaches:

- SAT solvers based on **stochastic search**: The solver first guesses a **full assignment** (also called **full valuation**), *i.e.*, an assignment of truth-values to all atoms. If the WFF evaluates to \( F \) under this assignment, it starts to flip truth-values of the atoms according to some heuristics. Typically, it counts the number of unsatisfied clauses and chooses the flip that minimizes this number.

- SAT solvers based on **exhaustive search**: The solver traverses a binary tree, in which internal nodes are **partial valuations** and leaves are **full valuations**, and repeatedly backtracks in search of a satisfying full valuation. Several sub-approaches:
  - DP (Davis, Putnam) procedure and many variations, based on **resolution**.\(^2\)
  - DPLL (Davis, Putnam, Logemann, Loveland) procedure and many variations, based on DP combined with “clever” **backtracking**.
  - CDCL (Conflict-Driven Clause Learning) procedures, partially based on DPLL and more “clever” ideas.

---

\(^2\)Search the Web for articles on “resolution” and “paramodulation” (a variation on resolution), sometimes in articles on general “automated theorem proving”.

Assaf Kfoury, CS 512, Spring 2016, Handout 07
Resolution Rule

- The rule is limited to propositional WFF's in CNF.
- The rule can be used by itself to establish the validity of an arbitrary CNF, as an alternative to a Natural-Deduction formal proof (Chapt. 1 in [LCS]).
Resolution Rule

- The rule is limited to propositional WFF's in CNF.
- The rule can be used by itself to establish the validity of an arbitrary CNF, as an alternative to a Natural-Deduction formal proof (Chapt. 1 in [LCS]).
- CNF clauses are each a disjunction of literals (atoms and negated atoms).
- The antecedents of the resolution rule are two clauses of a CNF:

\[
(\ell_1 \lor \cdots \lor \ell_{p-1} \lor \ell_p \lor \ell_{p+1} \cdots \lor \ell_m) \quad \text{and} \quad (\ell'_1 \lor \cdots \lor \ell'_{q-1} \lor \ell'_q \lor \ell'_{q+1} \cdots \lor \ell'_n)
\]

where all \( \ell_i \) and \( \ell'_j \) are literals, and \( \ell_p = \neg \ell'_q \).
Resolution Rule

- The rule is limited to propositional WFF’s in CNF.
- The rule can be **used by itself** to establish the validity of an arbitrary CNF, as an alternative to a Natural-Deduction formal proof (Chapt. 1 in [LCS]).
- CNF clauses are each a disjunction of literals (atoms and negated atoms).
- The antecedents of the resolution rule are two clauses of a CNF:
  \[(\ell_1 \lor \cdots \lor \ell_{p-1} \lor \ell_p \lor \ell_{p+1} \cdots \lor \ell_m) \quad \text{and} \quad (\ell'_1 \lor \cdots \lor \ell'_{q-1} \lor \ell'_q \lor \ell'_{q+1} \cdots \lor \ell'_n)\]
  where all \(\ell_i\) and \(\ell'_j\) are literals, and \(\ell_p = \neg \ell'_q\).
- The resolution rule applied to the pair \((\ell_p, \ell'_q)\) is:
  \[
  \begin{array}{l}
  (\ell_1 \lor \cdots \lor \ell_{p-1} \lor \ell_p \lor \ell_{p+1} \cdots \lor \ell_m) \\
  \ell_1 \lor \cdots \lor \ell_{p-1} \lor \ell_{p+1} \cdots \lor \ell_m \lor \ell'_1 \lor \cdots \lor \ell'_{q-1} \lor \ell'_q \lor \ell'_{q+1} \cdots \lor \ell'_n \\
  \hline
  \ell_1 \lor \cdots \lor \ell_{p-1} \lor \ell_{p+1} \cdots \lor \ell_m \lor \ell'_1 \lor \cdots \lor \ell'_{q-1} \lor \ell'_q \lor \ell'_{q+1} \cdots \lor \ell'_n
  \end{array}
  \]
  New clause produced by resolution (below the line) is the resolvent.
Resolution Rule: how to use it

First, a reminder: The Universe of Propositional WFF’s

- The negation of a tautology is an unsatisfiable WFF (Why? Look at the truth tables.)
- The negation of an unsatisfiable WFF is a tautology (Why? Look at the truth tables.)
- WFF’s that are neither tautologies nor contradictions are sometimes called contingent, which are in the middle of the diagram above.
Resolution Rule: how to use it

First, a reminder:

The Universe of Propositional WFF’s

- valid WFFs/tautologies
- unsatisfiable WFF’s/contradictions

▶ the negation of a tautology is an unsatisfiable WFF
(Why? Look at the truth tables.)

▶ the negation of an unsatisfiable WFF is a tautology
(Why? Look at the truth tables.)

▶ WFF’s that are neither tautologies nor contradictions are sometimes called contingent, which are in the middle of the diagram above.
Resolution Rule: how to use it

First, a reminder: The Universe of Propositional WFF’s

- The negation of a **tautology** is an **unsatisfiable WFF** (Why? Look at the truth tables.)
- The negation of an **unsatisfiable WFF** is a **tautology** (Why? Look at the truth tables.)
- WFF’s that are neither tautologies nor contradictions are sometimes called **contingent**, which are in the middle of the diagram above.
Resolution Rule: how to use it

First, a reminder: The Universe of Propositional WFF’s

- Valid WFFs/tautologies
- Unsatisfiable WFF’s/contradictions
- Satisfiable WFF’s

Warning: the negation of a satisfiable WFF is NOT an unsatisfiable WFF, even though the diagram and the words seem to suggest it!
Resolution Rule: how to use it

First, a reminder: **The Universe of Propositional WFF’s**

- **valid WFFs/tautologies**
- **unsatisfiable WFF's/contradictions**
- **satisfiable WFF’s**

▶ **Warning**: the negation of a *satisfiable WFF* is **NOT** an *unsatisfiable WFF*, even though the diagram and the words seem to suggest it!
Resolution Rule: how to use it

First, a reminder: The Universe of Propositional WFF’s

- **valid WFFs/tautologies**
- **unsatisfiable WFFs/contradictions**
- **falsifiable WFF's**

Warning: the negation of a falsifiable WFF is **NOT** a tautology, even though the diagram and the words seem to suggest it!
Resolution Rule: how to use it

First, a reminder: The Universe of Propositional WFF’s

- Valid WFFs/tautologies
- Falsifiable WFF's
- Unsatisfiable WFF's/contradictions

▶ Warning: the negation of a falsifiable WFF is NOT a tautology, even though the diagram and the words seem to suggest it!
Resolution Rule: how to use it

First, a reminder: **The Universe of Propositional WFF’s**

- **valid WFFs/tautologies**
- **unsatisfiable WFFs/contradictions**
- **satisfiable WFF’s**
- **falsifiable WFF’s**

Resolution can be used to decide membership of a propositional CNF in each of the 3 groups separately, but special care must be applied:

1. tautologies/valid WFF’s
2. contradictions/unsatisfiable WFF’s
3. contingent WFF’s, i.e., WFF’s that are both satisfiable and falsifiable.
Resolution Rule: how to use it

First, a reminder: \textbf{The Universe of Propositional WFF's}

- Valid WFFs/tautologies
- Unsatisfiable WFFs/contradictions
- Satisfiable WFFs
- Falsifiable WFFs

Resolution can be used to decide membership of a propositional CNF in each of the 3 groups separately, but special care must be applied:

1. Tautologies/valid WFF's
2. Contradictions/unsatisfiable WFF's
3. Contingent WFF's, i.e., WFF's that are both satisfiable and falsifiable
Resolution Rule: how to use it

- Before some examples, how strong is resolution?
Resolution Rule: how to use it

- Before some examples, how strong is resolution?
- Resolution is a sound and complete system of formal proofs for CNF’s, i.e., resolution is strong enough!
Resolution Rule: how to use it

▶ Before some examples, how strong is resolution?
▶ Resolution is a sound and complete system of formal proofs for CNF’s, i.e., resolution is strong enough!

From [LCS, Chapt 1], we already know:

Theorem
Let $\varphi$ be a propositional WFF. The following are equivalent statements:

1. $\varphi$ is valid, i.e., formally derivable.
2. $\varphi$ is a tautology, i.e., entries of last column of its truth-table are all $T$.
3. $\neg \varphi$ is a contradiction, i.e., $\bot$ is formally derivable from $\neg \varphi$.
4. $\neg \varphi$ is unsatisfiable, i.e., entries of last column of its truth-table are all $F$. 
Resolution Rule: how to use it

- Before some examples, how strong is resolution?
- Resolution is a sound and complete system of formal proofs for CNF’s, i.e., resolution is strong enough!

From [LCS, Chapt 1], we already know:

Theorem
Let \( \varphi \) be a propositional WFF. The following are equivalent statements:

1. \( \varphi \) is valid, i.e., formally derivable.
2. \( \varphi \) is a tautology, i.e., entries of last column of its truth-table are all \( T \).
3. \( \neg \varphi \) is a contradiction, i.e., \( \bot \) is formally derivable from \( \neg \varphi \).
4. \( \neg \varphi \) is unsatisfiable, i.e., entries of last column of its truth-table are all \( F \).

We specialize preceding theorem to CNF’s, but we must restrict it to parts 3 and 4 only:

Theorem
Let \( \psi \) be a propositional WFF in CNF. The following are equivalent statements:

3. \( \psi \) is a contradiction, i.e., \( \bot \) is formally derivable from \( \psi \) using resolution.
4. \( \psi \) is unsatisfiable, i.e., entries of last column of its truth-table are all \( F \).
Resolution Rule: how to use it

- Preceding theorem expresses soundness and completeness of resolution.
- Part 3 implies part 4 = **soundness** of resolution for CNF’s.
- Part 4 implies part 3 = **completeness** of resolution for CNF’s.
Resolution Rule: how to use it

- Preceding theorem expresses soundness and completeness of resolution.
- Part 3 implies part 4 = **soundness** of resolution for CNF’s.
- Part 4 implies part 3 = **completeness** of resolution for CNF’s.
- Observe carefully how **completeness** is used:
  1. From the two clauses \( \{p, q\} \), representing the CNF \( p \land q \), we can **NOT** apply **resolution** to formally derive \( p \lor q \), even though we know \( p, q \models p \lor q \).
     
     **So, how is it that resolution is said to be complete??**
  2. Completeness of another formal-proof system, such as **Natural Deduction**, means that
     \[
     \text{if } p, q \models p \lor q \text{ then } p, q \not\vdash p \lor q.
     \]
     **So, how is it that resolution is said to be complete??**
  3. However, from outside the **resolution**-based formal-proof system, \( i.e. \), at the meta-level, we know:
     \[
     \text{if } p, q \not\vdash p \lor q \text{ then } p, q \not\models p \lor q.
     \]
     which is just the converse of the usual implication of completeness.
     
     **So, how about using resolution to show if } p, q \not\vdash p \lor q \text{ then } p, q \not\models p \lor q??**
  4. Indeed, this is possible. At the meta-level, \( p, q \not\vdash p \lor q \) means the same thing as \( \{p, q, \neg(p \lor q)\} \) is **inconsistent** or **contradictory**, and **resolution** can derive this **inconsistency**, which will in turn imply \( p, q \models p \lor q \).

- This is why **resolution** is sometimes said to be **refutation-complete** rather than **complete**.
Resolution Rule: how to use it

Suppose we want to decide if propositional WFF $\psi$ is formally derivable from a given knowledge base, i.e., a finite set of premises: $\varphi_1, \ldots, \varphi_n$.

The following are the steps of a proof by contradiction to show that $\varphi_1, \ldots, \varphi_n \vdash \psi$, but note there is NO rule encoded into the formal derivation which is called PBC as in Natural Deduction:
Resolution Rule: how to use it

Suppose we want to decide if propositional WFF $\psi$ is formally derivable from a given knowledge base, i.e., a finite set of premises: $\varphi_1, \ldots, \varphi_n$.

The following are the steps of a proof by contradiction to show that $\varphi_1, \ldots, \varphi_n \vdash \psi$, but note there is NO rule encoded into the formal derivation which is called PBC as in Natural Deduction:

- Negate $\psi$ and add $\neg \psi$ to the knowledge base.
- Transform the knowledge base into a single CNF, thus obtaining a finite set of CNF clauses.
- Apply the resolution rule repeatedly, until there is no resolvable pair of clauses. (The procedure is bound to terminate – why?)
- Every time the resolution rule is applied, add the resolvent to the knowledge base.
- If $\bot$ (the empty clause) is produced, stop and report that the original $\psi$ is formally derivable from $\varphi_1, \ldots, \varphi_n$, i.e., $\varphi_1, \ldots, \varphi_n \vdash \psi$. 
Is the conjecture \( \neg P \) derivable from the knowledge base \( \{ P \rightarrow Q, Q \rightarrow R, \neg R \} \)?

- Negate the conjecture \( \neg \neg P = P \) and add it to the knowledge base.
- Transform all WFF’s in the knowledge base into CNF: \( \{ \neg P \lor Q, \neg Q \lor R, \neg R, P \} \).
- Putting down every clause in the knowledge base first, then applying the resolution rule repeatedly, we obtain:
Resolution Rule: small example

Is the conjecture $\neg P$ derivable from the knowledge base $\{P \rightarrow Q, Q \rightarrow R, \neg R\}$?

- Negate the conjecture $\neg \neg P = P$ and add it to the knowledge base.
- Transform all WFF's in the knowledge base into CNF: $\{\neg P \lor Q, \neg Q \lor R, \neg R, P\}$.
- Putting down every clause in the knowledge base first, then applying the resolution rule repeatedly, we obtain:

1. $\neg P \lor Q$
2. $\neg Q \lor R$
3. $\neg R$
4. $P$

▶ Stop and report that $\neg P$ is formally derivable from $\{P \rightarrow Q, Q \rightarrow R, \neg R\}$.
Resolution Rule: small example

Is the conjecture \( \neg P \) derivable from the knowledge base \( \{ P \rightarrow Q, Q \rightarrow R, \neg R \} \)?

▶ Negate the conjecture \( \neg \neg P = P \) and add it to the knowledge base.

▶ Transform all WFF’s in the knowledge base into CNF: \( \{ \neg P \lor Q, \neg Q \lor R, \neg R, P \} \).

▶ Putting down every clause in the knowledge base first, then applying the resolution rule repeatedly, we obtain:

\[
\begin{align*}
1 & \quad \neg P \lor Q \\
2 & \quad \neg Q \lor R \\
3 & \quad \neg R \\
4 & \quad P \\
5 & \quad Q & \text{resolve 1, 4} \\
6 & \quad R & \text{resolve 2, 5} \\
7 & \quad \bot & \text{resolve 3, 6}
\end{align*}
\]

▶ Stop and report that \( \neg P \) is formally derivable from \( \{ P \rightarrow Q, Q \rightarrow R, \neg R \} \).
Suppose we want to decide whether propositional WFF $\varphi$ is **satisfiable**.
The following are the steps of a procedure to decide satisfiability:
Resolution Rule: how to use it

Suppose we want to decide whether propositional WFF $\varphi$ is **satisfiable**.

The following are the steps of a procedure to decide satisfiability:

- Transform $\varphi$ into CNF, thus obtaining a finite set of clauses, the initial knowledge base.
- Apply the **resolution rule** repeatedly, until there is no resolvable pair of clauses. *(The procedure is bound to terminate – why?)*
- Every time the **resolution rule** is applied, add the **resolvent** to the knowledge base.
- If $\bot$ (the empty clause) is produced, stop and report that the original $\varphi$ is **unsatisfiable**.
- If there are no more resolvable pair of clauses (and $\bot$ is not produced), stop and report that the original $\varphi$ is **satisfiable**.
Resolution Rule: small example

Let \( \varphi := (q_1 \lor q_2 \lor q_3) \land (q_2 \lor \neg q_3 \lor \neg q_4) \land (\neg q_2 \lor q_5) \), which is already a CNF.

▶ Is \( \varphi \) satisfiable?
Resolution Rule: small example

Let $\varphi := (q_1 \lor q_2 \lor q_3) \land (q_2 \lor \neg q_3 \lor \neg q_4) \land (\neg q_2 \lor q_5)$, which is already a CNF.

- Is $\varphi$ satisfiable?
- Write down $\varphi$ as a set of clauses, the initial **knowledge base**:
  $$\{q_1 \lor q_2 \lor q_3, \ q_2 \lor \neg q_3 \lor \neg q_4, \ \neg q_2 \lor q_5\}.$$
- Put down every clause in the **knowledge base** first, then apply the resolution rule repeatedly, to obtain:
Resolution Rule: small example

Let \( \varphi := (q_1 \lor q_2 \lor q_3) \land (q_2 \lor \neg q_3 \lor \neg q_4) \land (\neg q_2 \lor q_5) \), which is already a CNF.

▶ Is \( \varphi \) satisfiable?

▶ Write down \( \varphi \) as a set of clauses, the initial knowledge base:
\[
\{ q_1 \lor q_2 \lor q_3, \ q_2 \lor \neg q_3 \lor \neg q_4, \ \neg q_2 \lor q_5 \}.
\]

▶ Put down every clause in the knowledge base first, then apply the resolution rule repeatedly, to obtain:

1. \( q_1 \lor q_2 \lor q_3 \)
2. \( q_2 \lor \neg q_3 \lor \neg q_4 \)
3. \( \neg q_2 \lor q_5 \)
Resolution Rule: small example

Let $\varphi := (q_1 \lor q_2 \lor q_3) \land (q_2 \lor \neg q_3 \lor \neg q_4) \land (\neg q_2 \lor q_5)$, which is already a CNF.

▶ Is $\varphi$ satisfiable?

▶ Write down $\varphi$ as a set of clauses, the initial knowledge base:

$\{q_1 \lor q_2 \lor q_3, \ q_2 \lor \neg q_3 \lor \neg q_4, \ \neg q_2 \lor q_5\}$.

▶ Put down every clause in the knowledge base first, then apply the resolution rule repeatedly, to obtain:

1. $q_1 \lor q_2 \lor q_3$
2. $q_2 \lor \neg q_3 \lor \neg q_4$
3. $\neg q_2 \lor q_5$
4. $q_1 \lor q_3 \lor q_5$ resolve 1, 3
5. $\neg q_3 \lor \neg q_4 \lor q_5$ resolve 2, 3
6. $q_1 \lor \neg q_4 \lor q_5$ resolve 4, 5

▶ there are no more resolvable pairs of clauses, stop and report $\varphi$ is satisfiable.
Resolution Rule: another small example

Let $\psi := (p_1 \lor p_2) \land (p_1 \lor \neg p_2) \land (\neg p_1 \lor p_3) \land (\neg p_1 \lor \neg p_3)$, already a CNF.

▶ Is $\psi$ satisfiable?
Resolution Rule: another small example

Let $\psi := (p_1 \lor p_2) \land (p_1 \lor \neg p_2) \land (\neg p_1 \lor p_3) \land (\neg p_1 \lor \neg p_3)$, already a CNF.

- Is $\psi$ satisfiable?

- Write down $\varphi$ as a set of clauses, the initial knowledge base:
  \[
  \{p_1 \lor p_2, \ p_1 \lor \neg p_2, \ \neg p_1 \lor p_3, \ \neg p_1 \lor \neg p_3\}.
  \]

- Put down every clause in the knowledge base first, then apply the resolution rule:
Resolution Rule: another small example

Let $\psi := (p_1 \lor p_2) \land (p_1 \lor \neg p_2) \land (\neg p_1 \lor p_3) \land (\neg p_1 \lor \neg p_3)$, already a CNF.

▶ Is $\psi$ satisfiable?

▶ Write down $\varphi$ as a set of clauses, the initial knowledge base:
   \{
   p_1 \lor p_2, \quad p_1 \lor \neg p_2, \quad \neg p_1 \lor p_3, \quad \neg p_1 \lor \neg p_3
   \}.

▶ Put down every clause in the knowledge base first, then apply the resolution rule:

1. $p_1 \lor p_2$
2. $p_1 \lor \neg p_2$
3. $\neg p_1 \lor p_3$
4. $\neg p_1 \lor \neg p_3$

▶ stop and report $\psi$ is unsatisfiable.
Resolution Rule: another small example

Let $\psi := (p_1 \lor p_2) \land (p_1 \lor \neg p_2) \land (\neg p_1 \lor p_3) \land (\neg p_1 \lor \neg p_3)$, already a CNF.

- Is $\psi$ satisfiable?
- Write down $\varphi$ as a set of clauses, the initial knowledge base:
  \[{p_1 \lor p_2}, {p_1 \lor \neg p_2}, {\neg p_1 \lor p_3}, {\neg p_1 \lor \neg p_3}\].
- Put down every clause in the knowledge base first, then apply the resolution rule:
  1. $p_1 \lor p_2$
  2. $p_1 \lor \neg p_2$
  3. $\neg p_1 \lor p_3$
  4. $\neg p_1 \lor \neg p_3$
  5. $p_1$ resolve 1, 2
  6. $p_3$ resolve 3, 5
  7. $\neg p_3$ resolve 4, 5
  8. $\bot$ resolve 6, 7

- stop and report $\psi$ is unsatisfiable.
Resolution Rule: improvements in using it

After each application of the resolution rule:

▶ Simple improvement: remove repeated literals in the resolvent.

▶ Simple improvement: if the two antecedent clauses contain each more than one literal, discard the antecedent clauses from knowledge base. In this case, if resolution is on the literal pair \((\ell_p, \ell'_q)\), a truth-value assignment satisfying the resolvent can always be extended to a truth-value assignment for \(\ell_p\) and \(\ell'_q\) that satisfies the antecedents.

▶ Simple improvement: if the resolvent contains complementary literals, discard the resolvent instead of adding it to knowledge base. In this case, the resolvent is a tautology, i.e., every assignment of truth-values satisfies it.

▶ Simple improvement: if resolution is on the literal pair \((\ell_p, \ell'_q)\), retain resolvent and discard every clause containing \(\ell_p\) or \(\ell'_q\)??

Almost . . . . What if a clause is just the literal \(\ell_p\) or \(\ell'_q\)?
Resolution Rule: improvements in using it

After each application of the resolution rule:

- **Simple improvement**: remove repeated literals in the resolvent.

- **Simple improvement**: if the two antecedent clauses contain each more than one literal, discard the antecedent clauses from knowledge base.
  
  In this case, if resolution is on the literal pair $\ell_p, \ell'_q$, a truth-value assignment satisfying the resolvent can always be extended to a truth-value assignment for $\ell_p$ and $\ell'_q$ that satisfies the antecedents.

- **Simple improvement**: if the resolvent contains complementary literals, discard the resolvent instead of adding it to knowledge base.
  
  In this case, the resolvent is a tautology, i.e., every assignment of truth-values satisfies it.

- **Simple improvement**: if resolution is on the literal pair $(\ell_p, \ell'_q)$, retain resolvent and discard every clause containing $\ell_p$ or $\ell'_q$?
  
  Almost . . . . What if a clause is just the literal $\ell_p$ or $\ell'_q$?

- **Advanced improvements**: see DPLL next . . .
Classical DPLL Procedure

- In the rest of this handout, we only refer to SAT solvers that are based on the DPLL procedure, because most modern SAT solvers are based on it.
- We formulate the Classical DPLL procedure as a transition system consisting of 5 transition rules, which are used to operate over a domain of states.
- The definition of a state and the full description of these 5 rules is on the two next slides.³

Classical DPLL Procedure: What Is a *Partial Valuation*?

Definition

- If $\varphi$ is a propositional WFF, then a *partial valuation* (or *model*) $M$ for $\varphi$ is an assignment of truth values *to some* (and possibly – but not necessarily – *to all*) the propositional atoms in $\varphi$.

- Example: If $\varphi := \neg (q_1 \lor \neg q_2) \land q_3$, then a partial valuation $M$ for $\varphi$ may be the sequence $\neg q_1 q_3$ meaning that $M$ assigns $F$ to $q_1$ and $T$ to $q_3$.

- Fact: In this example, $M$ can be extended to a total valuation that satisfies $\varphi$ by assigning $T$ to $q_2$.

- Another partial valuation $M'$ for $\varphi$ may be the sequence $q_1 q_3$ meaning that $M'$ assigns $T$ to both $q_1$ and $q_3$.

- Fact: $M'$ cannot be extended to a total valuation that satisfies $\varphi$. 
Classical DPLL Procedure: What Is a *Partial Valuation*?

Definition

- If $\varphi$ is a propositional WFF, then a *partial valuation* (or *model*) $M$ for $\varphi$ is an assignment of truth values *to some* (and possibly – but not necessarily – *to all*) the propositional atoms in $\varphi$.
- If $M$ is a partial valuation for $\varphi$, we write $M$ as a sequence of atoms or negated atoms occurring in $\varphi$. 
- Fact: In this example, $M$ can be extended to a total valuation that satisfies $\varphi$ by assigning $T$ to $q_2$.
- Another partial valuation $M'$ for $\varphi := \neg (q_1 \lor \neg q_2) \land q_3$ may be the sequence $q_1 q_3$ meaning that $M$ assigns $T$ to both $q_1$ and $q_3$.
- Fact: $M'$ cannot be extended to a total valuation that satisfies $\varphi$. 

Assaf Kloury, CS 512, Spring 2016, Handout 07
Definition

- If $\varphi$ is a propositional WFF, then a *partial valuation* (or *model*) $M$ for $\varphi$ is an assignment of truth values *to some* (and possibly – but not necessarily – *to all*) the propositional atoms in $\varphi$.
- If $M$ is a partial valuation for $\varphi$, we write $M$ as a sequence of atoms or negated atoms occurring in $\varphi$.
- For example, if $\varphi := \neg((q_1 \lor \neg q_2) \land q_3)$, then a partial valuation $M$ for $\varphi$ may be the sequence $\neg q_1 q_3$ meaning that $M$ assigns $F$ to $q_1$ and $T$ to $q_3$.

**Fact:** In this example, $M$ can be extended to a total valuation that satisfies $\varphi$ by assigning $T$ to $q_2$. 
Definition

- If \( \varphi \) is a propositional WFF, then a \textit{partial valuation} (or \textit{model}) \( \mathcal{M} \) for \( \varphi \) is an assignment of truth values \textit{to some} (and possibly – but not necessarily – \textit{to all}) the propositional atoms in \( \varphi \).
- If \( \mathcal{M} \) is a partial valuation for \( \varphi \), we write \( \mathcal{M} \) as a sequence of atoms or negated atoms occurring in \( \varphi \).
- For example, if \( \varphi := \neg((q_1 \lor \neg q_2) \land q_3) \), then a partial valuation \( \mathcal{M} \) for \( \varphi \) may be the sequence \( \neg q_1 \ q_3 \) meaning that \( \mathcal{M} \) assigns \( \text{T} \) to \( q_1 \) and \( \text{T} \) to \( q_3 \).
  
  \text{Fact:} \ If \ this \ example, \( \mathcal{M} \) can be extended to a total valuation that satisfies \( \varphi \) by assigning \( \text{T} \) to \( q_2 \).

- Another partial valuation \( \mathcal{M}' \) for \( \varphi := \neg((q_1 \lor \neg q_2) \land q_3) \) may be the sequence \( q_1 \ q_3 \) meaning that \( \mathcal{M} \) assigns \( \text{T} \) to both \( q_1 \) and \( q_3 \).
  
  \text{Fact:} \ \( \mathcal{M}' \) cannot be extended to a total valuation that satisfies \( \varphi \).
Definition

- A *state* in the Classical DPLL is a pair of the form $\mathcal{M} \parallel \varphi$ where $\varphi$ is a propositional WFF in CNF and $\mathcal{M}$ is a partial valuation for $\varphi$. 

For example, a state of Classical DPLL may look like:

$p_1 p_2 \neg q_1 \parallel \{p_2, \neg p_1 \lor \neg q_1, \neg p_1 \lor q_2, q_1 \lor \neg q_2 \lor p_1, \neg p_2 \lor p_1 \lor \neg q_3, \neg p_1 \lor p_2, q_3 \lor p_2\}$

where

- the partial valuation (left of "∥") assigns $T$ to $p_1$, $T$ to $p_2$, and $F$ to $q_1$,
- the CNF (right of "∥") is written as a set of 7 clauses.

Fact: In the preceding example, the partial valuation (left of "∥") can be extended to a total valuation, namely, "$p_1 p_2 \neg q_1 q_2$", that satisfies the CNF (right of "∥").
Classical DPLL Procedure: What Is a State?

Definition

- A state in the Classical DPLL is a pair of the form $\mathcal{M} \parallel \varphi$ where $\varphi$ is a propositional WFF in CNF and $\mathcal{M}$ is a partial valuation for $\varphi$.

- For example, a state of Classical DPLL may look like:

\[
p_1 \ p_2 \ \neg q_1 \parallel \{ \ p_2, \neg p_1 \lor \neg q_1, \neg p_1 \lor q_2, \ q_1 \lor \neg q_2 \lor p_1, \ \\
\quad \neg p_2 \lor p_1 \lor \neg q_3, \neg p_1 \lor p_2, \ q_3 \lor p_2 \ \}
\]

where

- the partial valuation (left of “$\parallel$”) assigns $T$ to $p_1$, $T$ to $p_2$, and $F$ to $q_1$,
- the CNF (right of “$\parallel$”) is written as a set of 7 clauses.
Classical DPLL Procedure: What Is a State?

Definition

- A state in the Classical DPLL is a pair of the form $\mathcal{M} \parallel \varphi$ where $\varphi$ is a propositional WFF in CNF and $\mathcal{M}$ is a partial valuation for $\varphi$.

- For example, a state of Classical DPLL may look like:

  $\begin{align*}
  p_1 & \quad p_2 \quad \neg q_1 \\
  \parallel & \quad \{ p_2, \neg p_1 \lor \neg q_1, \neg p_1 \lor q_2, q_1 \lor \neg q_2 \lor p_1, \\
  & \quad \neg p_2 \lor p_1 \lor \neg q_3, \neg p_1 \lor p_2, q_3 \lor p_2 \} 
  \end{align*}$

where

- the partial valuation (left of “$\parallel$”) assigns $T$ to $p_1$, $T$ to $p_2$, and $F$ to $q_1$,

- the CNF (right of “$\parallel$”) is written as a set of 7 clauses.

- **Fact**: In the preceding example, the partial valuation (left of “$\parallel$”) can be extended to a total valuation, namely, “$p_1 p_2 \neg q_1 q_2$”, that satisfies the CNF (right of “$\parallel$”).
Classical DPLL Procedure: 5 Transition Rules

**UnitPropagate**
\[ \mathcal{M} \parallel \varphi \cup \{ C \lor \ell \} \implies \mathcal{M} \ell \parallel \varphi \cup \{ C \lor \ell \} \]
if \( \mathcal{M} \models \neg C \) and \( \ell \) is undefined in \( \mathcal{M} \).

**PureLiteral**
\[ \mathcal{M} \parallel \varphi \implies \mathcal{M} \ell \parallel \varphi \]
if \( \ell \) occurs in a clause of \( \varphi \), \( \neg \ell \) occurs in no clause of \( \varphi \), and \( \ell \) is undefined in \( \mathcal{M} \).

**Decide**
\[ \mathcal{M} \parallel \varphi \implies \mathcal{M} \ell^d \parallel \varphi \]
if \( \ell \) or \( \neg \ell \) occurs in a clause of \( \varphi \) and \( \ell \) is undefined in \( \mathcal{M} \).

**Fail**
\[ \mathcal{M} \parallel \varphi \cup \{ C \} \implies \text{FailState} \]
if \( \mathcal{M} \models \neg C \) and \( \mathcal{M} \) contains no decision literals.

**Backtrack**
\[ \mathcal{M} \ell^d \mathcal{N} \parallel \varphi \cup \{ C \} \implies \mathcal{M} \neg \ell \parallel \varphi \cup \{ C \} \]
if \( \mathcal{M} \ell^d \mathcal{N} \models \neg C \) and \( \mathcal{N} \) contains no decision literals.
Classical DPLL Procedure: Example

Below is a derivation by Classical DPLL. For better readability:

- We denote the atoms $q_1, q_2, q_3, \ldots$ by their indices $1, 2, 3, \ldots$.
- We denote the negation $\neg q_k$ by $\overline{k}$.
Classical DPLL Procedure: Example

Below is a derivation by Classical DPLL. For better readability:

- We denote the atoms $q_1, q_2, q_3, \ldots$ by their indeces $1, 2, 3, \ldots$
- We denote the negation $\neg q_k$ by $\bar{k}$.

\[
\emptyset \parallel \bar{1} \lor 2, 2 \lor 3, \bar{1} \lor \bar{3} \lor 4, 2 \lor \bar{3} \lor \bar{4}, 1 \lor 4 \implies \text{(Decide)}
\]

\[
1^d \parallel \bar{1} \lor 2, 2 \lor 3, \bar{1} \lor \bar{3} \lor 4, 2 \lor \bar{3} \lor \bar{4}, 1 \lor 4 \implies \text{(UnitPropagate)}
\]

\[
1^d \bar{2} \parallel \bar{1} \lor 2, 2 \lor 3, \bar{1} \lor \bar{3} \lor 4, 2 \lor \bar{3} \lor \bar{4}, 1 \lor 4 \implies \text{(UnitPropagate)}
\]

\[
1^d \bar{2} 3 \parallel \bar{1} \lor 2, 2 \lor 3, \bar{1} \lor \bar{3} \lor 4, 2 \lor \bar{3} \lor \bar{4}, 1 \lor 4 \implies \text{(UnitPropagate)}
\]

\[
1^d \bar{2} 3 4 \parallel \bar{1} \lor 2, 2 \lor 3, \bar{1} \lor \bar{3} \lor 4, 2 \lor \bar{3} \lor \bar{4}, 1 \lor 4 \implies \text{(Backtrack)}
\]

\[
\bar{1} \parallel \bar{1} \lor 2, 2 \lor 3, \bar{1} \lor \bar{3} \lor 4, 2 \lor \bar{3} \lor \bar{4}, 1 \lor 4 \implies \text{(UnitPropagate)}
\]

\[
\bar{1} 4 \parallel \bar{1} \lor 2, 2 \lor 3, \bar{1} \lor \bar{3} \lor 4, 2 \lor \bar{3} \lor \bar{4}, 1 \lor 4 \implies \text{(Decide)}
\]

\[
\bar{1} 4 \bar{3}^d \parallel \bar{1} \lor 2, 2 \lor 3, \bar{1} \lor \bar{3} \lor 4, 2 \lor \bar{3} \lor \bar{4}, 1 \lor 4 \implies \text{(UnitPropagate)}
\]

\[
\bar{1} 4 \bar{3}^d \bar{2} \parallel \bar{1} \lor 2, 2 \lor 3, \bar{1} \lor \bar{3} \lor 4, 2 \lor \bar{3} \lor \bar{4}, 1 \lor 4
\]
Classical DPLL Procedure

Definition
A state $S$ is a **final state** if one of two conditions holds:

1. $S$ is the token “FailState”,
2. $S$ is of the form $\mathcal{M} \parallel \varphi$ where $\mathcal{M}$ is a total valuation for the CNF $\varphi$. 

Theorem
Let $\varphi$ be a WFF in CNF. Then:

1. Every derivation by Classical DPLL which starts with the state $\emptyset \parallel \varphi$ always terminates with a final state, i.e.:

   $\emptyset \parallel \varphi = \Rightarrow S_1 = \Rightarrow \cdots = \Rightarrow S_n$ for some $n \geq 1$ and where $S_n$ is a final state.

2. If the final state $S_n$ is of the form $\mathcal{M} \parallel \varphi$, then $\varphi$ is satisfiable and the total valuation $\mathcal{M}$ is a model of $\varphi$.

3. If the final state $S_n$ is the token “FailState”, then $\varphi$ is unsatisfiable.

Proof.
Left to the reader.
Classical DPLL Procedure

Definition
A state $S$ is a \textbf{final state} if one of two conditions holds:

1. $S$ is the token “FailState”,
2. $S$ is of the form $M \parallel \varphi$ where $M$ is a total valuation for the CNF $\varphi$.

Theorem
Let $\varphi$ be a WFF in CNF. Then:

1. Every derivation by Classical DPLL which starts with the state $\emptyset \parallel \varphi$ always terminates with a final state, i.e.:

$$\emptyset \parallel \varphi \Rightarrow S_1 \Rightarrow \cdots \Rightarrow S_n$$

for some $n \geq 1$ and where $S_n$ is a final state.
Classical DPLL Procedure

Definition
A state $S$ is a **final state** if one of two conditions holds:

1. $S$ is the token “FailState”,
2. $S$ is of the form $\mathcal{M} \parallel \varphi$ where $\mathcal{M}$ is a total valuation for the CNF $\varphi$.

Theorem
Let $\varphi$ be a WFF in CNF. Then:

1. Every derivation by Classical DPLL which starts with the state $\emptyset \parallel \varphi$ always terminates with a final state, i.e.:

   $\emptyset \parallel \varphi \Rightarrow S_1 \Rightarrow \cdots \Rightarrow S_n$

   for some $n \geq 1$ and where $S_n$ is a final state.

2. If the final state $S_n$ is of the form $\mathcal{M} \parallel \varphi$, then $\varphi$ is satisfiable and the total valuation $\mathcal{M}$ is a model of $\varphi$. 

Proof. Left to the reader.
Classical DPLL Procedure

Definition
A state $S$ is a **final state** if one of two conditions holds:

1. $S$ is the token “FailState”,
2. $S$ is of the form $M \parallel \varphi$ where $M$ is a total valuation for the CNF $\varphi$.

Theorem
Let $\varphi$ be a WFF in CNF. Then:

1. Every derivation by Classical DPLL which starts with the state $\emptyset \parallel \varphi$ always terminates with a final state, i.e.:

   $\emptyset \parallel \varphi \implies S_1 \implies \cdots \implies S_n$

   for some $n \geq 1$ and where $S_n$ is a final state.

2. If the final state $S_n$ is of the form $M \parallel \varphi$, then $\varphi$ is satisfiable and the total valuation $M$ is a model of $\varphi$.

3. If the final state $S_n$ is the token “FailState”, then $\varphi$ is unsatisfiable.

Proof.
Left to the reader.
Classical DPLL Procedure

**Definition**
A state $S$ is a **final state** if one of two conditions holds:

1. $S$ is the token “FailState”,
2. $S$ is of the form $\mathcal{M} \parallel \varphi$ where $\mathcal{M}$ is a total valuation for the CNF $\varphi$.

**Theorem**
Let $\varphi$ be a WFF in CNF. Then:

1. **Every derivation by Classical DPLL which starts with the state $\emptyset \parallel \varphi$ always terminates with a final state**, i.e.:

   \[
   \emptyset \parallel \varphi \implies S_1 \implies \cdots \implies S_n
   \]

   for some $n \geq 1$ and where $S_n$ is a final state.

2. **If the final state $S_n$ is of the form $\mathcal{M} \parallel \varphi$, then $\varphi$ is satisfiable and the total valuation $\mathcal{M}$ is a model of $\varphi$.**

3. **If the final state $S_n$ is the token “FailState”, then $\varphi$ is unsatisfiable.**

**Proof.**
Left to the reader. $\square$
Modern (DPLL-Based) SAT Solvers

- Modern SAT solvers are based on the classical DPLL procedure, in which they introduce several modifications for efficiency.

- Rule PureLiteral is used as a pre-processing step and excluded from the rules driving the solver, i.e., it is applied repeatedly before all other rules until it cannot be applied, after which it is not used.

- Rule Backtrack is replaced by a more general and powerful backtracking mechanism, the so-called Backjump rule.

- ... and other efficiency modifications.
Modern (DPLL-Based) SAT Solvers

- Modern SAT solvers are based on the classical DPLL procedure, in which they introduce several modifications for efficiency.
- Among the efficiencies introduced in modern SAT solvers:
  - Rule **PureLiteral** is used as a pre-processing step and excluded from the rules driving the solver, *i.e.*, it is applied repeatedly before all other rules until it cannot be applied, after which it is not used.
Modern (DPLL-Based) SAT Solvers

- Modern SAT solvers are based on the classical DPLL procedure, in which they introduce several modifications for efficiency.
- Among the efficiencies introduced in modern SAT solvers:
  - Rule **PureLiteral** is used as a pre-processing step and excluded from the rules driving the solver, *i.e.*, it is applied repeatedly before all other rules until it cannot be applied, after which it is not used.
  - Rule **Backtrack** is replaced by a more general and powerful backtracking mechanism, the so-called **Backjump** rule.
  - . . . and other efficiency modifications.
Modern (DPLL-Based) SAT Solvers

- A basic modern SAT solver omits the rule PureLiteral from the Classical DPLL procedure, but includes the 3 rules:

  UnitPropagate, Decide, and Fail (as before),

  together with (at least) the new rule Backjump instead of Backtrack.

\[ M \ell d N \parallel \phi \cup \{ C \} = \Rightarrow M \ell d' N \parallel \phi \cup \{ C \} \]

- Although Backjump is more efficient than Backtrack, it is a little more difficult to understand.

---

Modern (DPLL-Based) SAT Solvers

- A basic modern SAT solver omits the rule **PureLiteral** from the Classical DPLL procedure, but includes the 3 rules:
  
  **UnitPropagate**, **Decide**, and **Fail** (as before),

  together with (at least) the new rule **Backjump** instead of **Backtrack**.

- **Backjump**

  \[
  \mathcal{M} \ell^d \mathcal{N} \parallel \varphi \cup \{C\} \quad \Longrightarrow \quad \mathcal{M} \ell' \parallel \varphi \cup \{C\}
  \]

  if \( \mathcal{M} \ell^d \mathcal{N} \models \neg C \) and there is some clause \( C' \lor \ell' \) such that:

  1. \( \varphi \cup \{C\} \models C' \lor \ell' \),
  2. \( \mathcal{M} \models \neg C' \),
  3. \( \ell' \) is undefined in \( \mathcal{M} \), and
  4. \( \ell' \) or \( \neg \ell' \) occurs in \( \varphi \) or in \( \mathcal{M} \ell^d \mathcal{N} \).

Modern (DPLL-Based) SAT Solvers

- A basic modern SAT solver omits the rule PureLiteral from the Classical DPLL procedure, but includes the 3 rules:
  - UnitPropagate, Decide, and Fail (as before),
  - together with (at least) the new rule Backjump instead of Backtrack.

- **Backjump**

  \[ M \ell^d \mathcal{N} \parallel \varphi \cup \{C\} \implies M \ell' \parallel \varphi \cup \{C\} \]

  if \( M \ell^d \mathcal{N} \models \neg C \) and there is some clause \( C' \lor \ell' \) such that:
  1. \( \varphi \cup \{C\} \models C' \lor \ell' \),
  2. \( M \models \neg C' \),
  3. \( \ell' \) is undefined in \( M \), and
  4. \( \ell' \) or \( \neg \ell' \) occurs in \( \varphi \) or in \( M \ell^d \mathcal{N} \).

- Although Backjump is more efficient than Backtrack, it is a little more difficult to understand.\(^4\)

---

For efficiency, it turns out that Backjump works even better in the presence of two (non-essential, but more helpful for backtracking) rules:

*Forget* and *Learn*

---

Modern (DPLL-Based) SAT Solvers

For efficiency, it turns out that Backjump works even better in the presence of two (non-essential, but more helpful for backtracking) rules:

**Forget** and **Learn**

**Forget**

\[ M \parallel \varphi \cup \{C\} \implies M \parallel \varphi \]

if \( \varphi \models C \)

**Learn**

\[ M \parallel \varphi \implies M \parallel \varphi \cup \{C\} \]

if \( \varphi \models C \) and

each atom of \( C \) occurs in \( \varphi \) or in \( M \)

---

Modern (DPLL-Based) SAT Solvers

For efficiency, it turns out that **Backjump** works even better in the presence of two (non-essential, but more helpful for backtracking) rules:

**Forget** and **Learn**

**Forget**

\[
M \parallel \varphi \cup \{C\} \implies M \parallel \varphi \quad \text{if} \quad \varphi \models C
\]

**Learn**

\[
M \parallel \varphi \implies M \parallel \varphi \cup \{C\} \quad \text{if} \quad \varphi \models C \text{ and each atom of } C \text{ occurs in } \varphi \text{ or in } M
\]

Although the soundness of **Forget** and **Learn** is relatively easy to understand, *i.e.*, “their presence does not turn an unsatisfiable WFF into a satisfiable WFF,” it is more difficult to understand why they improve efficiency (in conjunction with **Backjump**).\(^5\)

SMT Solver = SAT Solver + a theory

- **SMT** = *Satisfiability Modulo a Theory*.
- **Theory** = *the quantifier-free fragment of a first-order theory*.\(^6\)
- **SMT Solver** = *SAT solver* working with a *theory solver* (or *T-solver*).

---

\(^6\) See Handout 08 for details on first-order theories.
SMT Solver = SAT Solver + a theory

- **SMT** = *Satisfiability Modulo a Theory*.
- **Theory** = *the quantifier-free fragment of a first-order theory*.\(^6\)
- **SMT Solver** = *SAT solver* working with a *theory solver* (or *T*-solver).

Examples of first-order theories considered in SMT solvers, in each case limited to the quantifier-free fragment:

- Equality with Uninterpreted Functions (EUF)
- Linear Integer Arithmetic (LIA)
- Linear Real Arithmetic (LRA)
- Difference Logic (DL), which is a fragment of LRA
- other theories:
  - Arrays, Bit-Vectors, Tuples and Records, Algebraic Datatypes, etc.

---

\(^6\) See Handout 08 for details on first-order theories.

Assaf Kfoury, CS 512, Spring 2016, Handout 07
SMT Solver = SAT Solver + a theory

- **SMT** = *Satisfiability Modulo a Theory*.
- **Theory** = *the quantifier-free fragment of a first-order theory*.\(^6\)
- **SMT Solver** = *SAT solver* working with a *theory solver* (or *T*-solver).

Examples of first-order theories considered in SMT solvers, in each case limited to the quantifier-free fragment:

- Equality with Uninterpreted Functions (EUF)
- Linear Integer Arithmetic (LIA)
- Linear Real Arithmetic (LRA)
- Difference Logic (DL), which is a fragment of LRA
- other theories:
  - Arrays, Bit-Vectors, Tuples and Records, Algebraic Datatypes, etc.

Reason for the restriction to quantifier-free fragments:
We need an efficient decision procedure to decide validity, i.e., to “quickly” decide whether \( T \vdash \varphi \).

---

\(^6\) See Handout 08 for details on first-order theories.
Two General Approaches to SMT Solving

▶ Eager Methods
▶ Convert SMT problem into an equisatisfiable SAT problem.
▶ Example theories for which eager methods work well:
  - Equality
  - Difference Logic
  - Bit-Vectors.

▶ Lazy Methods
▶ Interleave SAT -solver steps with T -solver steps, but keep the two separate.
▶ More widely applicable than eager methods.
▶ Most common approach:
  - CDCL SAT -solver combined with a T -solver.

7 CDCL SAT-solver = Conflict-Driven Clause-Learning SAT-solver, a variant of the DPLL procedure.
Two General Approaches to SMT Solving

▶ Eager Methods

▶ Lazy Methods

---

\[\text{CDCL SAT-solver} = \text{Conflict-Driven Clause-Learning SAT-solver},\]
\[\text{a variant of the DPLL procedure.}\]
Two General Approaches to SMT Solving

▶ Eager Methods

▶ Convert SMT problem into an equisatisfiable SAT problem.
▶ Example theories for which eager methods work well: Equality, Difference Logic, Bit-Vectors.

▶ Lazy Methods

---

7 CDCL SAT-solver = Conflict-Driven Clause-Learning SAT-solver, a variant of the DPLL procedure.
Two General Approaches to SMT Solving

▶ Eager Methods

▶ Convert SMT problem into an equisatisfiable SAT problem.
▶ Example theories for which eager methods work well: Equality, Difference Logic, Bit-Vectors.

▶ Lazy Methods

▶ Interleave SAT-solver steps with T-solver steps, but keep the two separate.
▶ More widely applicable than eager methods.
▶ Most common approach: CDCL SAT-solver combined with a T-solver.\(^7\)

\(^7\)CDCL SAT-solver = Conflict-Driven Clause-Learning SAT-solver, a variant of the DPLL procedure.
Resolution Rule: proof of soundness

Theorem
Let $\psi$ be a CNF, $\psi = \{C_1, \ldots, C_n\}$, where every clause $C_i$ is a finite disjunct of literals. Apply resolution repeatedly to $\Psi_0 = \psi$ to obtain the sequence of CNF's:

$$\Psi_0 \Psi_1 \Psi_2 \cdots \Psi_p$$

for some $p \geq 1$.

If $\bot \in \Psi_p$ then $\psi = \Psi_0$ is unsatisfiable.

(Leave aside whether the sequence is bound to terminate. Yes, it is bound to terminate!)
Resolution Rule: proof of soundness

Theorem
Let \( \psi \) be a CNF, \( \psi = \{C_1, \ldots, C_n\} \), where every clause \( C_i \) is a finite disjunct of literals. Apply resolution repeatedly to \( \Psi_0 = \psi \) to obtain the sequence of CNF’s:

\[
\Psi_0 \quad \Psi_1 \quad \Psi_2 \quad \cdots \quad \Psi_p \quad \text{for some } p \geq 1.
\]

If \( \bot \in \Psi_p \) then \( \psi = \Psi_0 \) is unsatisfiable.
(Leave aside whether the sequence is bound to terminate. Yes, it is bound to terminate!)

Proof.
Every time resolution is applied to some \( \Psi_i \), we have:

\[
\frac{(C \lor p) \land (D \lor \neg p)}{(C \lor D)}
\]

Resolvent \( C \lor D \) is satisfied by any truth-value assignment satisfying \( C \) or \( D \).
Hence, if \( \Psi_i \) is satisfiable, then so is \( \Psi_{i+1} = \Psi_i \cup \{(C \lor D)\} \).
Hence, resolution preserves satisfiability at every step from \( \Psi_0 \) to \( \Psi_p \).
Hence, if \( \Psi_p \) is unsatisfiable, then so is \( \Psi_0 \).
But \( \bot \in \Psi_p \) means \( \Psi_p \) is unsatisfiable, implying desired conclusion. \( \square \)
Resolution Rule: proof of completeness

Theorem
Let \( \psi \) be a CNF, \( \psi = \{C_1, \ldots, C_n\} \), where every clause \( C_i \) is a finite disjunct of literals. Apply resolution repeatedly to \( \Psi_0 = \psi \) to obtain the sequence of CNF’s:

\[
\Psi_0 \quad \Psi_1 \quad \Psi_2 \quad \cdots \quad \Psi_p \quad \text{for some } p \geq 1.
\]

If \( \psi = \Psi_0 \) is unsatisfiable, then \( \bot \in \Psi_p \).

(Leave aside whether the sequence is bound to terminate. Yes, it is bound to terminate!)
Resolution Rule: proof of completeness

**Theorem**

Let $\psi$ be a CNF, $\psi = \{C_1, \ldots, C_n\}$, where every clause $C_i$ is a finite disjunct of literals. Apply **resolution** repeatedly to $\Psi_0 = \psi$ to obtain the sequence of CNF's:

$$\Psi_0 \quad \Psi_1 \quad \Psi_2 \quad \cdots \quad \Psi_p$$

for some $p \geq 1$.

If $\psi = \Psi_0$ is unsatisfiable, then $\bot \in \Psi_p$.

(*Leave aside whether the sequence is bound to terminate. Yes, it is bound to terminate!*)

**Proof.**