Lecture 6
The DPLL Algorithm

Dr Christoph Haase
University of Oxford
(with small changes by Javier Esparza)
DPLL algorithm:

- combines search and deduction to decide satisfiability
- underlies most modern SAT solvers
- is over 50 years old
DPLL algorithm:
- combines search and deduction to decide satisfiability
- underlies most modern SAT solvers
- is over 50 years old

DPLL-based SAT solvers ≥ 1990:
- clause learning
- non-chronological backtracking
- branching heuristics
- lazy evaluation
Performance increase of SAT solvers

SAT/SMT solving improvements

#constraints solved in standard benchmark

year

DPLL: idea

Depth-first search.
At every unsuccessful leaf of search tree (called conflict),
use resolution to compute a conflict clause.
Add clause to formula we’re deciding about.

Think of conflict clauses as “caching” previous search results,
so we “learn from previous mistakes”.
Conflict clauses also determine backtracking.
The DPLL algorithm

**Input:** CNF formula $F$.

1. Initialise $\mathcal{A}$ to the empty assignment
2. While there is unit clause $\{L\}$ in $F|_{\mathcal{A}}$, update $\mathcal{A} \leftarrow \mathcal{A}[L \rightarrow 1]$
3. If $F|_{\mathcal{A}}$ contains no clauses, stop and output $\mathcal{A}$.
4. If $F|_{\mathcal{A}} \ni \Box$, add new clause $C$ to $F$ by learning procedure. If $C$ is the empty clause, stop and output UNSAT. Otherwise backtrack to highest level where $C$ is unit clause. Go to Line 2.
5. Apply decision strategy to update $\mathcal{A} \leftarrow \mathcal{A}[p \rightarrow b]$. Go to line 2.

$F|_{\mathcal{A}}$ is set of clauses obtained from deleting any clause containing true literal, and deleting from each remaining clause all false literals.
**Terminology**

**State** of algorithm is pair of CNF formula $F$ and assignment $\mathcal{A}$. **Successful state** when $\mathcal{A} \models F$. **Conflict state** when $\mathcal{A} \not\models F$. 
State of algorithm is pair of CNF formula $F$ and assignment $A$. Successful state when $A \models F$. Conflict state when $A \not\models F$.

- Each assignment $p_i \mapsto b_i$ classifies as decision assignment or implied assignment.
Terminology

State of algorithm is pair of CNF formula $F$ and assignment $A$. Successful state when $A \models F$. Conflict state when $A \not\models F$.

- Each assignment $p_i \mapsto b_i$ classifies as decision assignment or implied assignment.
- $p_i$ in a decision assignment $p_i \mapsto b_i$ is decision variable.
**Terminology**

**State** of algorithm is pair of CNF formula $F$ and assignment $\mathcal{A}$. **Successful state** when $\mathcal{A} \models F$. **Conflict state** when $\mathcal{A} \not\models F$.

- Each assignment $p_i \mapsto b_i$ classifies as **decision assignment** or **implied assignment**.
- $p_i$ in a decision assignment $p_i \mapsto b_i$ is **decision variable**.
- Denote by $p_i \overset{C}{\mapsto} b_i$ an implied assignment arising through **unit propagation** on clause $C$.

(Note: conflict state if $F \models \mathcal{A} \not\models \Box$, successful state if $F \models \mathcal{A} = \emptyset$.)
**Terminology**

- **State** of algorithm is pair of CNF formula $F$ and assignment $\mathcal{A}$. **Successful state** when $\mathcal{A} \models F$. **Conflict state** when $\mathcal{A} \not\models F$.

- Each assignment $p_i \mapsto b_i$ classifies as decision assignment or implied assignment.

- $p_i$ in a decision assignment $p_i \mapsto b_i$ is decision variable.

- Denote by $p_i \overset{C}{\mapsto} b_i$ an implied assignment arising through unit propagation on clause $C$.

- **Decision level** of assignment $p_i \mapsto b_i$ in a given state $\mathcal{A}$ is number of decision assignments in $\mathcal{A}$ that precede $p_i \mapsto b_i$. (Note: conflict state if $F \models \mathcal{A} \ni \Box$, successful state if $F \models \mathcal{A} = \emptyset$)
**Terminology**

**State** of algorithm is pair of CNF formula $F$ and assignment $\mathcal{A}$. **Successful state** when $\mathcal{A} \models F$. **Conflict state** when $\mathcal{A} \not\models F$.

- Each assignment $p_i \mapsto b_i$ classifies as **decision assignment** or **implied assignment**.
- $p_i$ in a decision assignment $p_i \mapsto b_i$ is **decision variable**.
- Denote by $p_i \overset{C}{\mapsto} b_i$ an implied assignment arising through **unit propagation** on clause $C$.
- **Decision level** of assignment $p_i \mapsto b_i$ in a given state $\mathcal{A}$ is number of decision assignments in $\mathcal{A}$ that precede $p_i \mapsto b_i$.

(Note: conflict state if $F|_\mathcal{A} \ni \Box$, successful state if $F|_\mathcal{A} = \emptyset$)
Unit propagation

Unit propagation: the while loop in line 2 updates assignment $L \mapsto 1$ whenever there is unit clause $\{L\} \in F|_A$. 

Example: start with set of clauses $F = \{C_1, \ldots, C_5\}$, where

$C_1 = \{\neg p_1, \neg p_4, p_5\}$

$C_2 = \{\neg p_1, p_6, \neg p_5\}$

$C_3 = \{\neg p_1, \neg p_6, p_7\}$

$C_4 = \{\neg p_1, \neg p_7, \neg p_5\}$

$C_5 = \{p_1, p_4, p_6\}$

Say current assignment is $(p_1 \mapsto 1, p_2 \mapsto 0, p_3 \mapsto 0, p_4 \mapsto 1)$. Notice $F|_A$ contains unit clause $\{p_5\}$. Unit propagation further generates $(p_5 \mapsto 1, p_6 \mapsto 1, p_7 \mapsto 1)$. This leads to a conflict, with $C_4$ being made false.
Unit propagation: the while loop in line 2 updates assignment \( L \mapsto 1 \) whenever there is unit clause \( \{L\} \in F|_A \).

Example: start with set of clauses \( F = \{C_1, \ldots, C_5\} \), where

\[
\begin{align*}
C_1 &= \{\neg p_1, \neg p_4, p_5\} \\
C_2 &= \{\neg p_1, p_6, \neg p_5\} \\
C_3 &= \{\neg p_1, \neg p_6, p_7\} \\
C_4 &= \{\neg p_1, \neg p_7, \neg p_5\} \\
C_5 &= \{p_1, p_4, p_6\}
\end{align*}
\]

Say current assignment is \((p_1 \mapsto 1, p_2 \mapsto 0, p_3 \mapsto 0, p_4 \mapsto 1)\). Notice \( F|_A \) contains unit clause \( \{p_5\} \).

Unit propagation further generates \((p_5 \overset{C_1}{\mapsto} 1, p_6 \overset{C_2}{\mapsto} 1, p_7 \overset{C_3}{\mapsto} 1)\). This leads to a conflict, with \( C_4 \) being made false.
Conflict analysis

After unit propagation:
- If not in conflict nor successful, make decision (line 5)
- If in conflict, **learned clause** is added (line 4)
Conflict analysis

After unit propagation:
- If not in conflict nor successful, make decision (line 5)
- If in conflict, **learned clause** is added (line 4)

**Learned clause desiderata:** If unit propagation from state \((F, \mathcal{A})\) leads to conflict, clause \(C\) is learned such that:
Conflict analysis

After unit propagation:
- If not in conflict nor successful, make decision (line 5)
- If in conflict, learned clause is added (line 4)

Learned clause desiderata: If unit propagation from state \((F, \mathcal{A})\) leads to conflict, clause \(C\) is learned such that:

1. \(F \equiv F \cup \{C\}\)
Conflicts analysis

After unit propagation:
- If not in conflict nor successful, make decision (line 5)
- If in conflict, **learned clause** is added (line 4)

**Learned clause desiderata**: If unit propagation from state \((F, \mathcal{A})\) leads to conflict, clause \(C\) is learned such that:

1. \(F \equiv F \cup \{C\}\)
2. \(C\) is **conflict clause**: each literal is made false by \(\mathcal{A}\)
Conflict analysis

After unit propagation:
- If not in conflict nor successful, make decision (line 5)
- If in conflict, **learned clause** is added (line 4)

**Learned clause desiderata:** If unit propagation from state \((F, A)\) leads to conflict, clause \(C\) is learned such that:

1. \(F \equiv F \cup \{C\}\)
2. \(C\) is **conflict clause**: each literal is made false by \(A\)
3. \(C\) mentions only decision variables in \(A\)
Clause learning

Suppose $A = (p_1 \leftrightarrow b_1, \ldots, p_k \leftrightarrow b_k)$ leads to conflict. Find associated clauses $A_1, \ldots, A_{k+1}$ by backward induction:
Clause learning

Suppose $\mathcal{A} = (p_1 \rightarrow b_1, \ldots, p_k \rightarrow b_k)$ leads to conflict. Find associated clauses $A_1, \ldots, A_{k+1}$ by backward induction:

1. Take any conflict clause under $\mathcal{A}$ as $A_{k+1}$
Clause learning

Suppose $A = (p_1 \mapsto b_1, \ldots, p_k \mapsto b_k)$ leads to conflict. Find associated clauses $A_1, \ldots, A_{k+1}$ by backward induction:

1. Take any conflict clause under $A$ as $A_{k+1}$

2. If $p_i \mapsto b_i$ is decision assignment or $p_i$ not mentioned in $A_{i+1}$, set $A_i = A_{i+1}$
Clause learning

Suppose $A = (p_1 \mapsto b_1, \ldots, p_k \mapsto b_k)$ leads to conflict. Find associated clauses $A_1, \ldots, A_{k+1}$ by backward induction:

1. Take any conflict clause under $A$ as $A_{k+1}$

2. If $p_i \mapsto b_i$ is decision assignment or $p_i$ not mentioned in $A_{i+1}$, set $A_i = A_{i+1}$

3. If $p_i \overset{C_i}{\mapsto} b_i$ is implied assignment and $p_i$ mentioned in $A_{i+1}$, define $A_i$ to be resolvent of $A_{i+1}$ and $C_i$ with respect to $p_i$
Clause learning

Suppose $A = (p_1 \mapsto b_1, \ldots, p_k \mapsto b_k)$ leads to conflict. Find associated clauses $A_1, \ldots, A_{k+1}$ by backward induction:

1. Take any conflict clause under $A$ as $A_{k+1}$

2. If $p_i \mapsto b_i$ is decision assignment or $p_i$ not mentioned in $A_{i+1}$, set $A_i = A_{i+1}$

3. If $p_i \overrightarrow{C_i} b_i$ is implied assignment and $p_i$ mentioned in $A_{i+1}$, define $A_i$ to be resolvent of $A_{i+1}$ and $C_i$ with respect to $p_i$

The final clause $A_1$ is the **learned clause**
Clause learning: example

In conflict of above example, learning generates clauses

\[ A_8 := \{ \neg p_1, \neg p_7, \neg p_5 \} \]  \hspace{1cm} \text{(clause } C_4\text{)}
\[ A_7 := \{ \neg p_1, \neg p_5, \neg p_6 \} \]  \hspace{1cm} \text{(resolve } A_8, C_3\text{)}
\[ A_6 := \{ \neg p_1, \neg p_5 \} \]  \hspace{1cm} \text{(resolve } A_7, C_2\text{)}
\[ A_5 := \{ \neg p_1, \neg p_4 \} \]  \hspace{1cm} \text{(resolve } A_6, C_1\text{)}
\[ \vdots \]
\[ A_1 := \{ \neg p_1, \neg p_4 \} \]
Clause learning: example

In conflict of above example, learning generates clauses

\[
A_8 := \{\neg p_1, \neg p_7, \neg p_5\} \quad \text{(clause } C_4) \\
A_7 := \{\neg p_1, \neg p_5, \neg p_6\} \quad \text{(resolve } A_8, C_3) \\
A_6 := \{\neg p_1, \neg p_5\} \quad \text{(resolve } A_7, C_2) \\
A_5 := \{\neg p_1, \neg p_4\} \quad \text{(resolve } A_6, C_1) \\
\vdots \\
A_1 := \{\neg p_1, \neg p_4\}
\]

Learned clause \(A_1\) is conflict clause with only decision variables, including top-level one \(p_4\). Intuitively:

- record that conflict arose from decision to make \(p_1, p_4\) true
- adding \(A_1\) makes assignments validating \(p_1, p_4\) unreachable
- backtrack to highest level where \(A_1\) is unit clause (\(p_1 \mapsto 1\), unit propagation leads to \(p_4 \mapsto 0\).
Clause learning

**Proposition:** this policy fulfills the desiderata

**Proof sketch:** *Observation:* If \( p_i \overset{C_i}{\rightarrow} b_i \), then the only literal of \( C_i \) true under \( \mathcal{A} \) is the literal for \( p_i \) (that is, \( C_i \) contains either \( p_i \) or \( \neg p_i \), and \( b_i \) is chosen to make the literal true).
Clause learning

**Proposition:** this policy fulfills the desiderata

**Proof sketch:** Observation: If $p_i \overset{C_i}{ightarrow} b_i$, then the only literal of $C_i$ true under $\mathcal{A}$ is the literal for $p_i$ (that is, $C_i$ contains either $p_i$ or $\neg p_i$, and $b_i$ is chosen to make the literal true).

1. $F \equiv F \cup \{C\}$
   Because $C$ is obtained from clauses of $F$ through resolution steps.
Clause learning

Proposition: this policy fulfills the desiderata

Proof sketch: Observation: If \( p_i \overset{C_i}{\rightarrow} b_i \), then the only literal of \( C_i \) true under \( \mathcal{A} \) is the literal for \( p_i \) (that is, \( C_i \) contains either \( p_i \) or \( \neg p_i \), and \( b_i \) is chosen to make the literal true).

1. \[ F \equiv F \cup \{ C \} \]
Because \( C \) is obtained from clauses of \( F \) through resolution steps.

2. \( C \) is conflict clause: each literal is made false by \( \mathcal{A} \).
We show by induction that it holds for \( A_{k+1}, A_k, A_{k-1} \cdots A_1 = C \).
It holds for \( A_{k+1} \) by definition.
If it holds for \( A_{i+1} \) and \( A_i = A_{i+1} \), then obviously it holds for \( A_i \).
If it holds for \( A_{i+1} \) and \( A_i \neq A_{i+1} \), then \( A_i \) is the result of resolving \( A_{i+1} \) and \( C_i \). By the observation, all literals of \( A_i \) are made false by \( \mathcal{A} \).
Clause learning

Proposition: this policy fulfills the desiderata

Proof sketch: Observation: If \( p_i \overset{C_i}{\rightarrow} b_i \), then the only literal of \( C_i \) true under \( \mathcal{A} \) is the literal for \( p_i \) (that is, \( C_i \) contains either \( p_i \) or \( \neg p_i \), and \( b_i \) is chosen to make the literal true).

1. \( F \equiv F \cup \{ C \} \)
   Because \( C \) is obtained from clauses of \( F \) through resolution steps.

2. \( C \) is conflict clause: each literal is made false by \( \mathcal{A} \).
   We show by induction that it holds for \( A_{k+1}, A_k, A_{k-1} \cdots A_1 = C \).
   It holds for \( A_{k+1} \) by definition.
   If it holds for \( A_{i+1} \) and \( A_i = A_{i+1} \), then obviously it holds for \( A_i \).
   If it holds for \( A_{i+1} \) and \( A_i \neq A_{i+1} \), then \( A_i \) is the result of resolving \( A_{i+1} \) and \( C_i \). By the observation, all literals of \( A_i \) are made false by \( \mathcal{A} \).

3. \( C \) mentions only decision variables in \( \mathcal{A} \).
   Because every other variable, say \( p_i \), dissapears after resolving with \( A_{i+1} \) w.r.t. \( p_i \). Indeed, since \( \mathcal{A} \) makes \( A_{i+1} \) false, by the observation \( p_i \) has opposite signs in \( A_{i+1} \) and \( C_i \).
Example: 4 queens

Problem: place 4 non-attacking queens on a 4x4 chess board
Example: 4 queens

Problem: place 4 non-attacking queens on a 4x4 chess board

Variable $p_{ij}$ models: there is a queen in square $(i, j)$

- $\geq 1$ in each row: $\bigwedge_{i=1}^{4} \bigvee_{j=1}^{4} p_{ij}$

- $\leq 1$ in each row: $\bigwedge_{i=1}^{4} \bigwedge_{j \neq j'}^{4} \neg p_{ij} \lor \neg p_{ij'}$

- $\leq 1$ in each column: $\bigwedge_{j=1}^{4} \bigwedge_{i \neq i'}^{4} \neg p_{ij} \lor \neg p_{i'j}$

- $\leq 1$ on each diagonal: $\bigwedge_{i,j=1}^{4} \bigvee_{k} \neg p_{i-k, i+k} \lor \neg p_{i+k, j+k}$

Total number of clauses: $4 + 24 + 24 + 28 = 80$
DPLL: 4 queens

Running the DPLL algorithm:

- Start with $p_{11} \rightarrow 1$
  - delete $\{p_{11}, p_{12}, p_{13}, p_{14}\}$, delete $\neg p_{11}$: 9 new unit clauses
  - unit propagation: deletes 65 clauses!

  - Set $p_{23} \rightarrow 1$
    - 4 new unit clauses: $\{\neg p_{24}\}, \{\neg p_{43}\}, \{\neg p_{32}\}, \{\neg p_{34}\}$
    - unit propagation of $\{\neg p_{34}\}$: \textsc{UNSAT}

  - fixing only two literals collapsed from 80 clauses to 1
    - ruled out 2 of 14 possible assignments!

  - Backtrack: $p_{11} \rightarrow 0, p_{12} \rightarrow 1$
    - delete $\{\neg p_{12}\}$: 9 new unit clauses
    - unit propagation: leaves only 1 clause $\{p_{43}\}$

Answer: $p_{12}, p_{24}, p_{31}, p_{43} \rightarrow 1$
DPLL: 4 queens

Running the DPLL algorithm:

- Start with $p_{11} \mapsto 1$
  delete $\{p_{11}, p_{12}, p_{13}, p_{14}\}$, delete $\neg p_{11}$: 9 new unit clauses
  unit propagation: deletes 65 clauses!

- Set $p_{23} \mapsto 1$
  4 new unit clauses: $\{\neg p_{24}\}, \{\neg p_{43}\}, \{\neg p_{32}\}, \{\neg p_{34}\}$
  unit propagation of $\{\neg p_{34}\}$: UNSAT
DPLL: 4 queens

Running the DPLL algorithm:

- Start with $p_{11} \mapsto 1$
  delete $\{p_{11}, p_{12}, p_{13}, p_{14}\}$, delete $\neg p_{11}$: 9 new unit clauses
  unit propagation: deletes 65 clauses!

- Set $p_{23} \mapsto 1$
  4 new unit clauses: $\\{\neg p_{24}\}, \\{\neg p_{43}\}, \\{\neg p_{32}\}, \\{\neg p_{34}\}$
  unit propagation of $\\{\neg p_{34}\}$: UNSAT
  fixing only two literals collapsed from 80 clauses to 1
  ruled out $2^{14}$ of $2^{16}$ possible assignments!

- Backtrack: $p_{11} \mapsto 0$, $p_{12} \mapsto 1$
  delete $\\{\neg p_{12}\}$: 9 new unit clauses
  unit propagation: leaves only 1 clause $\{p_{43}\}$!
DPLL: 4 queens

Running the DPLL algorithm:

- Start with $p_{11} \rightarrow 1$
  delete $\{p_{11}, p_{12}, p_{13}, p_{14}\}$, delete $\neg p_{11}$: 9 new unit clauses
  unit propagation: deletes 65 clauses!

- Set $p_{23} \rightarrow 1$
  4 new unit clauses: $\{\neg p_{24}\}, \{\neg p_{43}\}, \{\neg p_{32}\}, \{\neg p_{34}\}$
  unit propagation of $\{\neg p_{34}\}$: UNSAT
  fixing only two literals collapsed from 80 clauses to 1
  ruled out $2^{14}$ of $2^{16}$ possible assignments!

- Backtrack: $p_{11} \rightarrow 0, p_{12} \rightarrow 1$
  delete $\{\neg p_{12}\}$: 9 new unit clauses
  unit propagation: leaves only 1 clause $\{p_{43}\}$!

- Answer: $p_{12}, p_{24}, p_{31}, p_{43} \rightarrow 1$
Summary

- **Resolution:**
  - very simple sound and complete proof calculus
  - basis for type unification

- **DPLL algorithm**
  - improves resolution with clause learning and backtracking
  - very efficient basis for modern SAT solvers