This week we’ll study CDCL, the state-of-the-art algorithm for SAT solving. CDCL extends DPLL to learn derived clauses that capture the “root cause” of conflicts, allowing the solver to generalize conflicts to “lemmas” which help cull the state space.

As we will see, these learned clauses (lemmas) are added in a “sound” way that do not change satisfiability, but help accelerate the search.

The CDCL algorithm has a lot of moving parts, and really takes some time to understand—we will spend a few days discussing.
Last Lecture: DPLL

Main idea: search-based algorithm for materializing an enumeration. Very space efficient, but time? Leaves a lot to be desired. Early-day computers are space pressured.

\[
\begin{align*}
\neg A \lor B \lor C \\
A \lor C \lor D \\
A \lor C \lor \neg D \\
A \lor \neg C \lor D \\
\neg B \lor \neg C \lor D \\
\neg A \lor B \lor \neg C \\
\neg A \lor \neg B \lor C
\end{align*}
\]

\[
\begin{align*}
A &= \text{True} \\
B &= \text{False} \\
C &= \text{True} \\
D &= \text{True}
\end{align*}
\]

Luckily, the forced unit propagation $D=\text{True}$ leads us to success!
The basic structure of CDCL is similar to DPLL:

- Perform **unit propagation** (i.e., “forced assignments”) until it is not possible to do so anymore.
- If you hit a contradiction in doing just unit propagation then you return UNSAT—no guessing was involved.
- Now, in a loop, guess an assignment:
  - **Decide** an undecided variable.
  - Unit propagate to a fixed point (until you can’t anymore).
  - When you reach a conflict:
    - **Learn** clause that captures the “reason” for the conflict.
    - **Backtrack** non-chronologically in a way informed by the conflict, to avoid reaching the same conflict again.
Reminder: Unit Propagation

Unit propagation (boolean constraint propagation) is the idea that (unsatisfied) clauses containing only one unassigned literal are “forced” to have that value in any satisfying assignment.

For example, consider the clause $x_3 \lor \neg x_5 \lor \neg x_7 \lor x_8$, and the current partial assignment $x_3 = F, x_5 = T, x_8 = F$, now $\neg x_7$ is a logical consequence of the current valuation whenever the clause holds.

Similarly, if there is another clause: $\neg x_5 \lor x_7 \lor x_9$, unit propagation of $\neg x_7$ now forces the assignment of $x_9$ (as everything else if false).
Decision Levels

DPLL and CDCL both **alternate between BCP and decision (guessing)**

Building on the intuition from the last slide, it is sensible to construe the “decision level” as the number of *decision literals* in a trace.

So when we refer to the “decision level” of the CDCL algorithm, we are saying: how many **genuine guesses** have we made up until this point—ignoring the (possibly large) number of forced implications due to iterative application of BCP.
Exercise: Boolean Constraint Propagation

In sum, boolean constraint propagation performs one-step transitive reasoning to derived forced implications.

For the following SAT instance, show how iterated application of unit propagation will lead to a conflict:

\begin{align*}
(1) & \neg x_3 \lor x_5 \lor x_6 \\
(2) & x_3 \lor x_5 \\
(3) & \neg x_5 \\
(4) & \neg x_6
\end{align*}
Exercise Solution...

One possible solution: $\neg x_5$ forces $x_5 = \text{False}$. Now perform BCP: this obtains $x_3$ from (2) and thus $x_3 = \text{True}$. Now, from (4) conclude $x_6 = \text{False}$. BCP of $\neg x_6$ on (1) yields $\neg x_3'$, which gives a conflict with the previously-derived $x_3 = \text{True}$ (from BCP of 2 on $\neg x_5$).
The trail

A key data structure in the CDCL algorithm is the notion of a **trail**, a LIFO list of decisions along side their provenance. The trail accumulates a partial assignment and captures (a part of) the solver’s execution at each step in the computation.

The trail is a list of literals, each of these literals is annotated with either (a) a special **dec** token, indicating that the reason for the literal’s inclusion was a decision, (b) a unit clause present in the instance, or (c) a pointer to the clause that forced propagation via BCP.

The trail is useful because it separates the solver’s decision points from its unit propagations. Later on, we will backtrack to decision points, throwing away the unit propagations after a (wrong) decision.
The trail records the “focus” into the tree of DPLL, but separates (a) possibly-bad guesses (decision nodes) and (b) forced consequences of those decisions.

One of CDCL’s key insights is that the decisions in the (a) category are the ones we really need to be tuning—after we choose (a), (b) is inevitable!
The trail

When we write a trail, we may also include the decision level via “@N” where N is the decision level. Thus, our trails will have the form

\[ l_1@0, \ldots, l_k^{\text{dec}}, l_j@1, \ldots, l_{k+1}^{\text{dec}}, l_{j+1}@2, \ldots \]

I.e., an initial set of unit clauses and their propagations, followed by decisions with (possibly zero) consequences of unit propagation.

Conflicts from BCP at level 0 represent UNSAT problems that require **no guessing to prove UNSAT**

These represent “easy” UNSAT instances, no guessing involved!
Example trail

(1) \( \neg x_3 \lor x_5 \lor x_6 \)
(2) \( x_3 \lor x_5 \)
(3) \( \neg x_5 \)
(4) \( \neg x_6 \)

For the problem here, the trail looks like the following:

\( \neg x_5 \oplus 0^{(3)}, x_3 \oplus 0^{(2)}, \neg x_6 \oplus 0^{(4)}, \neg x_3 \oplus 0^{(1)}, \text{CONFLICT} \)
Now an example which is satisfiable

Consider the following clauses:

1. $x_0 \lor x_3 \lor \neg x_4$
2. $\neg x_3 \lor x_5$
3. $x_4 \lor x_3$

No unit clauses, and thus we must guess. We are guessing at **decision-level zero**, and we will guess $x_3 = \text{True}$, our trail looks like $x_3 @0^{\text{dec}}$
Now an example which is satisfiable

Consider the following clauses:
(1) \( x_0 \lor x_3 \lor \neg x_4 \)
(2) \( \neg x_3 \lor x_5 \)
(3) \( x_4 \lor x_3 \)

Now, we need to perform BCP for \( x_3 \), we always eagerly apply BCP. Now (2) tells us we need to decide \( x_5 \), and our trail looks like:

\( x_3 \oplus 0^{\text{dec}}, x_5 \oplus 1^{(2)} \)
Now an example which is satisfiable

Consider the following clauses:
(1) \( x_0 \lor x_3 \lor \neg x_4 \)
(2) \( \neg x_3 \lor x_5 \)
(3) \( x_4 \lor x_3 \)

Now we can’t apply BCP anymore (all clauses satisfied), we have unassigned values, but their values are arbitrary (1/2/3 already satisfied):

\( x_3 @ 0^{\text{dec}}, x_5 @ 1^{(2)}, x_0 @ 1^{\text{dec}}, \neg x_4 @ 2^{\text{dec}} \)

This gives us a satisfying assignment: \( x_0 = T, x_3 = T, x_5 = T, x_4 = F \)
Algorithm 1 Typical CDCL algorithm

```
CDCL(\varphi, \nu)
1   if (UnitPropagation(\varphi, \nu) == CONFLICT)
2       then return UNSAT
3   \text{dl} \leftarrow 0 \quad \triangleright \text{Decision level}
4   \text{while (not AllVariablesAssigned(\varphi, \nu))}
5       do \text{((x, v) \leftarrow PickBranchingVariable(\varphi, \nu)) \quad \triangleright \text{Decide stage}}
6       \text{dl} \leftarrow \text{dl} + 1 \quad \triangleright \text{Increment decision level due to new decision}
7       \nu \leftarrow \nu \cup \{(x, v)\}
8       \text{if (UnitPropagation(\varphi, \nu) == CONFLICT) \quad \triangleright \text{Deduce stage}}
9           \text{then } \beta = ConflictAnalysis(\varphi, \nu) \quad \triangleright \text{Diagnose stage}
10          \text{if } (\beta < 0)
11              \text{then return UNSAT}
12          \text{else Backtrack(\varphi, \nu, \beta)}
13         \text{dl} \leftarrow \beta \quad \triangleright \text{Decrement decision level due to backtracking}
14   return SAT
```

[Joao Marques-Silva, Ines Lynce and Sharad Malik
From the “Handbook of Satisfiability]
Implication Graph

Solvers work in terms of linear trails, but it is semantically useful to construe an *implication graph*, whose vertices are literals and whose edges define the “forced implications” from the rules.
Step 1 — Decide

No possible unit propagation, thus decide
Arbitrarily, we decide ¬x₁

Implication Graph

New node in implication graph, all root nodes are decisions. Non-root nodes are results of BCP
(Bottom right of node labels decision level)
Step 2 — BCP

First clause forces $x_4$, extend trail rooted at $x_1$

Implication Graph

New non-root node $x_4$
Step 3 — Decide Again

\[
\begin{align*}
&x_1 \lor x_4 \\
&x_1 \lor \neg x_3 \lor \neg x_8 \\
&x_1 \lor \neg x_8 \lor x_{12} \\
&x_2 \lor x_{11} \\
&\neg x_7 \lor \neg x_3 \lor x_9 \\
&\neg x_7 \lor x_9 \\
&\neg x_7 \lor x_8 \lor \neg x_9 \\
&x_7 \lor x_8 \lor \neg x_{10} \\
&x_7 \lor x_{10} \lor \neg x_{12}
\end{align*}
\]

Still more unassigned literals—keep going, next let’s assign \(x_3\) (True). This is a new decision level.
Step 4 — More BCP

\[ x_1 \lor x_4 \]
\[ x_1 \lor \neg x_3 \lor \neg x_8 \]
\[ x_1 \lor \neg x_8 \lor x_{12} \]
\[ x_2 \lor x_{11} \]
\[ \neg x_7 \lor \neg x_3 \lor x_9 \]
\[ \neg x_7 \lor x_8 \lor \neg x_9 \]
\[ x_7 \lor x_8 \lor \neg x_{10} \]
\[ x_7 \lor x_{10} \lor \neg x_{12} \]

Now \( x_3 \) is true and \( x_1 \) is false, thus BCP \( \neg x_8 \)

Implication Graph

\( \neg x_1 \)  \( \neg x_1 \)@0, \( x_4 \)@0

\( x_3 \)  \( x_3 \)@1, \( \neg x_8 \)@1
Step 5 — Even More BCP

\[ x_1 \lor x_4 \]
\[ x_1 \lor \neg x_3 \lor \neg x_9 \]
\[ x_1 \lor \neg x_8 \lor x_{12} \]
\[ x_2 \lor x_{11} \]
\[ \neg x_7 \lor \neg x_3 \lor x_9 \]
\[ \neg x_7 \lor x_8 \lor \neg x_9 \]
\[ x_7 \lor x_8 \lor \neg x_{10} \]
\[ x_7 \lor x_{10} \lor \neg x_{12} \]

We can now infer \( x_{12} \) from \( \neg x_8 \)

Implication Graph
Step 6 — Back to Guessing

\[ x_1 \vee x_4 \]
\[ x_1 \vee \neg x_3 \vee \neg x_8 \]
\[ x_1 \vee \neg x_8 \vee x_{12} \]
\[ x_2 \vee x_{11} \]
\[ \neg x_7 \vee \neg x_3 \vee x_9 \]
\[ \neg x_7 \vee x_8 \vee \neg x_9 \]
\[ x_7 \vee x_8 \vee \neg x_{10} \]
\[ x_7 \vee x_{10} \vee \neg x_{12} \]

Implication Graph

Now let’s guess $\neg x_2$

\[ \neg x_1 \rightarrow x_1@0, x_4@0 \]
\[ x_3@1, \neg x_8@1, x_{12}@1 \]
\[ \neg x_2 \rightarrow x_2@2 \]
Step 7 — BCP

We’re now forced to decide $x_{11}$

Implication Graph
Step 8 — Guess yet Again!

\[
\begin{align*}
\neg x_1 & \lor x_4 \\
\neg x_1 & \lor \neg x_3 \lor \neg x_8 \\
\neg x_1 & \lor \neg x_8 \lor x_{12} \\
\neg x_2 & \lor x_{11} \\
\neg x_7 & \lor \neg x_3 \lor x_9 \\
\neg x_7 & \lor x_8 \lor \neg x_9 \\
x_7 & \lor x_8 \lor \neg x_{10} \\
x_7 & \lor x_{10} \lor \neg x_{12}
\end{align*}
\]

Still no answer, let’s try \( x_7 \)

Implication Graph

\[ \neg x_1 \quad \neg x_1@0, x_4@0 \]
\[ x_3 \quad x_3@1, \neg x_8@1, x_{12}@1 \]
\[ \neg x_2 \quad \neg x_2@2, x_{11}@2 \]
\[ x_7 \quad x_7@3 \]
Step 9 — BCP & Conflict

After deciding $x_7$, we apply BCP

These two clauses yield a conflict

Implication Graph
Step 10 — Analyze Conflict

We’re at a conflict, we need to (a) decide on new “learned” clause and (b) decide where to backjump.

Implication Graph
Step 10 — Analyze Conflict

Idea: “cut out” the conflict

Identify cut in implication graph which separates latest decision node and conflict

Implication Graph

-x1, -x1@0, x4@0
x3@1, -x8@1, x12@1
-x2, -x2@2, x11@2
x7@3

-x9

x3
-x8

x7

x12

x1

x11

x4

x9

x7
Step 10 — Analyze Conflict

The “reason” is the incoming nodes sitting along the boundary

Together, these nodes form a **sufficient condition** for the conflict.
Step 10 — Analyze Conflict

Thus, to "explain" the conflict we can assert \( \neg(x_3 \land \neg x_8 \land x_7) \)

i.e., \( \neg x_3 \lor x_8 \lor \neg x_7 \)

i.e., \( \neg x_3 \lor x_8 \lor \neg x_7 \)
Step 11 — Backtrack

\[ x_1 \lor x_4 \]
\[ x_1 \lor \neg x_3 \lor \neg x_9 \]
\[ x_1 \lor \neg x_8 \lor x_{12} \]
\[ x_2 \lor x_{11} \]
\[ \neg x_7 \lor \neg x_3 \lor x_9 \]
\[ \neg x_7 \lor x_6 \lor \neg x_9 \]
\[ x_7 \lor x_8 \lor \neg x_{10} \]
\[ x_7 \lor x_{10} \lor \neg x_{12} \]

Implication Graph

Now we “backtrack” to \( x_3 \)
Non-Chronological Backtracking

In DPLL, we backtrack “one level.”
In CDCL, we backtrack to the second most recent decision level in the conflict clause. Or, equivalently, backtrack to the highest decision level in the conflict clause other than the most recent decision level.