CS 357: Advanced Topics in Formal Methods
Fall 2019

Lecture 4

Aleksandar Zeljić
(materials by Clark Barrett)
Stanford University

Abstract DPLL

We now return to DPLL. To facilitate a deeper look at DPLL, we use a high-level framework called *Abstract DPLL*.

Abstract DPLL

We now return to DPLL. To facilitate a deeper look at DPLL, we use a high-level framework called *Abstract DPLL*.

- Abstract DPLL uses *states* and *transitions* to model the progress of the algorithm.

Abstract DPLL

We now return to DPLL. To facilitate a deeper look at DPLL, we use a high-level framework called *Abstract DPLL*.

- Abstract DPLL uses *states* and *transitions* to model the progress of the algorithm.
- Most states are of the form $M \parallel F$, where
  - M is a *sequence of* annotated *literals* denoting a partial truth assignment, and
  - F is the CNF formula being checked, represented as a *set of clauses*.

Abstract DPLL

We now return to DPLL. To facilitate a deeper look at DPLL, we use a high-level framework called *Abstract DPLL*.

- Abstract DPLL uses *states* and *transitions* to model the progress of the algorithm.
- Most states are of the form $M \parallel F$, where
  - M is a *sequence of* annotated *literals* denoting a partial truth assignment, and
  - F is the CNF formula being checked, represented as a *set of clauses*.
- The *initial state* is $\emptyset \parallel F$, where F is to be checked for satisfiability.

Abstract DPLL

We now return to DPLL. To facilitate a deeper look at DPLL, we use a high-level framework called *Abstract DPLL*.

- Abstract DPLL uses *states* and *transitions* to model the progress of the algorithm.
- Most states are of the form $M \parallel F$, where
  - M is a *sequence of* annotated *literals* denoting a partial truth assignment, and
  - F is the CNF formula being checked, represented as a *set of clauses*.
- The *initial state* is $\emptyset \parallel F$, where F is to be checked for satisfiability.
- Transitions between states are defined by a set of *conditional transition rules*.

Abstract DPLL

The *final state* is either:

- a special fail state: $fail$, if F is unsatisfiable, or
- $M \parallel G$, where G is a CNF formula equisatisfiable with the original formula F, and M satisfies G

We write $M \models C$ to mean that for every truth assignment v, $v(M) = True$ implies $v(C) = True$.

Abstract DPLL Rules

UnitProp :

$$M \parallel F,\, C \vee l \quad \Longrightarrow \quad M\, l \parallel F,\, C \vee l \quad \textbf{if} \begin{cases} M \models \neg C \\ l \text{ is undefined in } M \end{cases}$$

Abstract DPLL Rules

UnitProp :

$$M \parallel F, C \vee l \quad \Longrightarrow \quad M\, l \parallel F, C \vee l \quad \textbf{if} \begin{cases} M \models \neg C \\ l \text{ is undefined in } M \end{cases}$$

PureLiteral :

$$M \parallel F \quad \Longrightarrow \quad M\, l \parallel F \quad \textbf{if} \begin{cases} l \text{ occurs in some clause of } F \\ \neg l \text{ occurs in no clause of } F \\ l \text{ is undefined in } M \end{cases}$$

Abstract DPLL Rules

UnitProp :

$$M \parallel F, C \vee l \quad \Longrightarrow \quad M\ l \parallel F, C \vee l \quad \textbf{if} \begin{cases} M \models \neg C \\ l \text{ is undefined in } M \end{cases}$$

PureLiteral :

$$M \parallel F \quad \Longrightarrow \quad M\ l \parallel F \quad \textbf{if} \begin{cases} l \text{ occurs in some clause of } F \\ \neg l \text{ occurs in no clause of } F \\ l \text{ is undefined in } M \end{cases}$$

Decide :

$$M \parallel F \quad \Longrightarrow \quad M\ l^{\mathrm{d}} \parallel F \quad \textbf{if} \begin{cases} \mathrm{l} \text{ or } \neg \mathrm{l} \text{ occurs in a clause of } F \\ \mathrm{l} \text{ is undefined in } \mathrm{M} \end{cases}$$

Abstract DPLL Rules

UnitProp :

$$M \parallel F,\, C \vee l \quad\Longrightarrow\quad M\, l \parallel F,\, C \vee l \quad \textbf{if} \begin{cases} M \models \neg C \\ l \text{ is undefined in } M \end{cases}$$

PureLiteral :

$$M \parallel F \quad\Longrightarrow\quad M\, l \parallel F \quad \textbf{if} \begin{cases} l \text{ occurs in some clause of } F \\ \neg l \text{ occurs in no clause of } F \\ l \text{ is undefined in } M \end{cases}$$

Decide :

$$M \parallel F \quad\Longrightarrow\quad M\, l^{\mathrm{d}} \parallel F \quad \textbf{if} \begin{cases} \mathrm{l} \text{ or } \neg \mathrm{l} \text{ occurs in a clause of } F \\ \mathrm{l} \text{ is undefined in } \mathrm{M} \end{cases}$$

Backtrack :

$$M\, l^{\mathrm{d}}\, N \parallel F,\, C \quad\Longrightarrow\quad M\, \neg l \parallel F,\, C \quad \textbf{if} \begin{cases} M\, l^{\mathrm{d}}\, N \models \neg C \\ \mathrm{N} \text{ contains no decision literals} \end{cases}$$

Abstract DPLL Rules

UnitProp :

$$M \parallel F,\ C \vee l \quad \Longrightarrow \quad M\ l \parallel F,\ C \vee l \quad \textbf{if} \begin{cases} M \models \neg C \\ l \text{ is undefined in } M \end{cases}$$

PureLiteral :

$$M \parallel F \quad \Longrightarrow \quad M\ l \parallel F \quad \textbf{if} \begin{cases} l \text{ occurs in some clause of } F \\ \neg l \text{ occurs in no clause of } F \\ l \text{ is undefined in } M \end{cases}$$

Decide :

$$M \parallel F \quad \Longrightarrow \quad M\ l^{\mathrm{d}} \parallel F \quad \textbf{if} \begin{cases} \mathrm{l} \text{ or } \neg \mathrm{l} \text{ occurs in a clause of } F \\ \mathrm{l} \text{ is undefined in } \mathrm{M} \end{cases}$$

Backtrack :

$$M\ l^{\mathrm{d}}\ N \parallel F,\ C \quad \Longrightarrow \quad M\ \neg l \parallel F,\ C \quad \textbf{if} \begin{cases} M\ l^{\mathrm{d}}\ N \models \neg C \\ \mathrm{N} \text{ contains no decision literals} \end{cases}$$

Fail :

$$M \parallel F,\ C \quad \Longrightarrow \quad \mathit{fail} \quad \textbf{if} \begin{cases} M \models \neg C \\ \mathrm{M} \text{ contains no decision literals} \end{cases}$$

Example

$$\emptyset \parallel \quad 1 \vee \bar{2},\ \ \bar{1} \vee \bar{2},\ \ 2 \vee 3,\ \ \bar{3} \vee 2,\ \ 1 \vee 4$$

Example

$$\begin{array}{lllllllll} \emptyset & \| & 1 \vee \overline{2}, & \overline{1} \vee \overline{2}, & 2 \vee 3, & \overline{3} \vee 2, & 1 \vee 4 & \Longrightarrow & (\mathsf{PureLiteral}) \\ 4 & \| & 1 \vee \overline{2}, & \overline{1} \vee \overline{2}, & 2 \vee 3, & \overline{3} \vee 2, & 1 \vee 4 & & \end{array}$$

Example

$$\begin{array}{rcllllllll}
\emptyset & \| & 1 \vee \overline{2}, & \overline{1} \vee \overline{2}, & 2 \vee 3, & \overline{3} \vee 2, & 1 \vee 4 & \Longrightarrow & \text{(PureLiteral)} \\
4 & \| & 1 \vee \overline{2}, & \overline{1} \vee \overline{2}, & 2 \vee 3, & \overline{3} \vee 2, & 1 \vee 4 & \Longrightarrow & \text{(Decide)} \\
4\ 1^{d} & \| & 1 \vee \overline{2}, & \overline{1} \vee \overline{2}, & 2 \vee 3, & \overline{3} \vee 2, & 1 \vee 4 & &
\end{array}$$

Example

$$\begin{array}{rcllllllcl}
\emptyset & \| & 1 \vee \overline{2}, & \overline{1} \vee \overline{2}, & 2 \vee 3, & \overline{3} \vee 2, & 1 \vee 4 & \Longrightarrow & \text{(PureLiteral)} \\
4 & \| & 1 \vee \overline{2}, & \overline{1} \vee \overline{2}, & 2 \vee 3, & \overline{3} \vee 2, & 1 \vee 4 & \Longrightarrow & \text{(Decide)} \\
4\ 1^{d} & \| & 1 \vee \overline{2}, & \overline{1} \vee \overline{2}, & 2 \vee 3, & \overline{3} \vee 2, & 1 \vee 4 & \Longrightarrow & \text{(UnitProp)} \\
4\ 1^{d}\ \overline{2} & \| & 1 \vee \overline{2}, & \overline{1} \vee \overline{2}, & 2 \vee 3, & \overline{3} \vee 2, & 1 \vee 4 & &
\end{array}$$

Example

$$
\begin{array}{rcllllllll}
\emptyset & \| & 1 \vee \overline{2}, & \overline{1} \vee \overline{2}, & 2 \vee 3, & \overline{3} \vee 2, & 1 \vee 4 & \Longrightarrow & (\text{PureLiteral}) \\
4 & \| & 1 \vee \overline{2}, & \overline{1} \vee \overline{2}, & 2 \vee 3, & \overline{3} \vee 2, & 1 \vee 4 & \Longrightarrow & (\text{Decide}) \\
4\ 1^d & \| & 1 \vee \overline{2}, & \overline{1} \vee \overline{2}, & 2 \vee 3, & \overline{3} \vee 2, & 1 \vee 4 & \Longrightarrow & (\text{UnitProp}) \\
4\ 1^d\ \overline{2} & \| & 1 \vee \overline{2}, & \overline{1} \vee \overline{2}, & 2 \vee 3, & \overline{3} \vee 2, & 1 \vee 4 & \Longrightarrow & (\text{UnitProp}) \\
4\ 1^d\ \overline{2}\ 3 & \| & 1 \vee \overline{2}, & \overline{1} \vee \overline{2}, & 2 \vee 3, & \overline{3} \vee 2, & 1 \vee 4 & &
\end{array}
$$

Example

$$
\begin{array}{rcl}
\emptyset \parallel 1 \vee \overline{2},\ \overline{1} \vee \overline{2},\ 2 \vee 3,\ \overline{3} \vee 2,\ 1 \vee 4 & \Longrightarrow & \text{(PureLiteral)} \\
4 \parallel 1 \vee \overline{2},\ \overline{1} \vee \overline{2},\ 2 \vee 3,\ \overline{3} \vee 2,\ 1 \vee 4 & \Longrightarrow & \text{(Decide)} \\
4\ 1^{d} \parallel 1 \vee \overline{2},\ \overline{1} \vee \overline{2},\ 2 \vee 3,\ \overline{3} \vee 2,\ 1 \vee 4 & \Longrightarrow & \text{(UnitProp)} \\
4\ 1^{d}\ \overline{2} \parallel 1 \vee \overline{2},\ \overline{1} \vee \overline{2},\ 2 \vee 3,\ \overline{3} \vee 2,\ 1 \vee 4 & \Longrightarrow & \text{(UnitProp)} \\
4\ 1^{d}\ \overline{2}\ 3 \parallel 1 \vee \overline{2},\ \overline{1} \vee \overline{2},\ 2 \vee 3,\ \overline{3} \vee 2,\ 1 \vee 4 & \Longrightarrow & \text{(Backtrack)} \\
4\ \overline{1} \parallel 1 \vee \overline{2},\ \overline{1} \vee \overline{2},\ 2 \vee 3,\ \overline{3} \vee 2,\ 1 \vee 4 & &
\end{array}
$$

Example

$\emptyset$	$\parallel$	$1\lor\overline{2}$,	$\overline{1}\lor\overline{2}$,	$2\lor 3$,	$\overline{3}\lor 2$,	$1\lor 4$	$\Longrightarrow$	(PureLiteral)
4	$\parallel$	$1\lor\overline{2}$,	$\overline{1}\lor\overline{2}$,	$2\lor 3$,	$\overline{3}\lor 2$,	$1\lor 4$	$\Longrightarrow$	(Decide)
$4\ 1^{d}$	$\parallel$	$1\lor\overline{2}$,	$\overline{1}\lor\overline{2}$,	$2\lor 3$,	$\overline{3}\lor 2$,	$1\lor 4$	$\Longrightarrow$	(UnitProp)
$4\ 1^{d}\ \overline{2}$	$\parallel$	$1\lor\overline{2}$,	$\overline{1}\lor\overline{2}$,	$2\lor 3$,	$\overline{3}\lor 2$,	$1\lor 4$	$\Longrightarrow$	(UnitProp)
$4\ 1^{d}\ \overline{2}\ 3$	$\parallel$	$1\lor\overline{2}$,	$\overline{1}\lor\overline{2}$,	$2\lor 3$,	$\overline{3}\lor 2$,	$1\lor 4$	$\Longrightarrow$	(Backtrack)
$4\ \overline{1}$	$\parallel$	$1\lor\overline{2}$,	$\overline{1}\lor\overline{2}$,	$2\lor 3$,	$\overline{3}\lor 2$,	$1\lor 4$	$\Longrightarrow$	(UnitProp)
$4\ \overline{1}\ \overline{2}\ \overline{3}$	$\parallel$	$1\lor\overline{2}$,	$\overline{1}\lor\overline{2}$,	$2\lor 3$,	$\overline{3}\lor 2$,	$1\lor 4$		

Example

$$
\begin{array}{rcllllllc l}
\emptyset & \| & 1 \lor \overline{2}, & \overline{1} \lor \overline{2}, & 2 \lor 3, & \overline{3} \lor 2, & 1 \lor 4 & & \Longrightarrow & (\mathsf{PureLiteral}) \\
4 & \| & 1 \lor \overline{2}, & \overline{1} \lor \overline{2}, & 2 \lor 3, & \overline{3} \lor 2, & 1 \lor 4 & & \Longrightarrow & (\mathsf{Decide}) \\
4\ 1^{\mathsf{d}} & \| & 1 \lor \overline{2}, & \overline{1} \lor \overline{2}, & 2 \lor 3, & \overline{3} \lor 2, & 1 \lor 4 & & \Longrightarrow & (\mathsf{UnitProp}) \\
4\ 1^{\mathsf{d}}\ \overline{2} & \| & 1 \lor \overline{2}, & \overline{1} \lor \overline{2}, & 2 \lor 3, & \overline{3} \lor 2, & 1 \lor 4 & & \Longrightarrow & (\mathsf{UnitProp}) \\
4\ 1^{\mathsf{d}}\ \overline{2}\ 3 & \| & 1 \lor \overline{2}, & \overline{1} \lor \overline{2}, & 2 \lor 3, & \overline{3} \lor 2, & 1 \lor 4 & & \Longrightarrow & (\mathsf{Backtrack}) \\
4\ \overline{1} & \| & 1 \lor \overline{2}, & \overline{1} \lor \overline{2}, & 2 \lor 3, & \overline{3} \lor 2, & 1 \lor 4 & & \Longrightarrow & (\mathsf{UnitProp}) \\
4\ \overline{1}\ \overline{2}\ \overline{3} & \| & 1 \lor \overline{2}, & \overline{1} \lor \overline{2}, & 2 \lor 3, & \overline{3} \lor 2, & 1 \lor 4 & & \Longrightarrow & (\mathsf{Fail}) \\
 & \mathit{fail} & & & & & & & &
\end{array}
$$

Example

$$
\begin{array}{rcl}
\emptyset \parallel & 1\vee\overline{2},\ \overline{1}\vee\overline{2},\ 2\vee 3,\ \overline{3}\vee 2,\ 1\vee 4 & \Longrightarrow \quad (\text{PureLiteral}) \\
4 \parallel & 1\vee\overline{2},\ \overline{1}\vee\overline{2},\ 2\vee 3,\ \overline{3}\vee 2,\ 1\vee 4 & \Longrightarrow \quad (\text{Decide}) \\
4\ 1^{d} \parallel & 1\vee\overline{2},\ \overline{1}\vee\overline{2},\ 2\vee 3,\ \overline{3}\vee 2,\ 1\vee 4 & \Longrightarrow \quad (\text{UnitProp}) \\
4\ 1^{d}\ \overline{2} \parallel & 1\vee\overline{2},\ \overline{1}\vee\overline{2},\ 2\vee 3,\ \overline{3}\vee 2,\ 1\vee 4 & \Longrightarrow \quad (\text{UnitProp}) \\
4\ 1^{d}\ \overline{2}\ 3 \parallel & 1\vee\overline{2},\ \overline{1}\vee\overline{2},\ 2\vee 3,\ \overline{3}\vee 2,\ 1\vee 4 & \Longrightarrow \quad (\text{Backtrack}) \\
4\ \overline{1} \parallel & 1\vee\overline{2},\ \overline{1}\vee\overline{2},\ 2\vee 3,\ \overline{3}\vee 2,\ 1\vee 4 & \Longrightarrow \quad (\text{UnitProp}) \\
4\ \overline{1}\ \overline{2}\ \overline{3} \parallel & 1\vee\overline{2},\ \overline{1}\vee\overline{2},\ 2\vee 3,\ \overline{3}\vee 2,\ 1\vee 4 & \Longrightarrow \quad (\text{Fail}) \\
\textit{fail} & &
\end{array}
$$

Result: *Unsatisfiable*

Abstract DPLL: Backjumping and Learning

The basic rules can be improved by replacing the Backtrack rule with the more powerful Backjump rule and adding a Learn rule:

Backjump :

$$M\ l^{\mathrm{d}}\ N \parallel F,\ C \quad \Longrightarrow \quad M\ l' \parallel F,\ C \quad \textbf{if} \begin{cases} M\ l^{\mathrm{d}}\ N \models \neg C, \text{ and there is} \\ \text{some clause } C' \vee l' \text{ such that :} \\ \ F, C \models C' \vee l' \text{ and } M \models \neg C', \\ \ l' \text{ is undefined in } \mathrm{M}, \text{ and} \\ \ \mathrm{l}' \text{ or } \neg \mathrm{l}' \text{ occurs in } F \text{ or in } M\ l^{\mathrm{d}}\ N \end{cases}$$

Learn :

$$M \parallel F \quad \Longrightarrow \quad M \parallel F,\ C \quad \textbf{if} \begin{cases} \text{all atoms of } C \text{ occur in } F \\ F \models C \end{cases}$$

Abstract DPLL: Backjumping and Learning

The Backjump rule is best understood by introducing the notion of *implication graph*, a directed graph associated with a state $M \parallel F$ of Abstract DPLL:

- The vertices are the *variables* in M
- There is an edge from v_1 to v_2 if v_2 was assigned a value as the result of an application of UnitProp using a clause containing v_2.

When we reach a state in which $M \models \neg C$ for some $C \in F$, we add an extra *conflict* vertex and edges from each of the variables in C to the conflict vertex.

Abstract DPLL: Backjumping and Learning

The clause to use for backjumping (called the *conflict clause*) is obtained from the resulting graph:

- We first cut the graph along edges in such a way that it separates the conflict vertex from all of the decision vertices.
- Then, every vertex with an outgoing edge that was cut is marked.
- For each literal l in M whose variable is marked, $\neg l$ is added to the conflict clause.

To avoid ever having the same conflict again, we can learn the conflict clause using the *learn* rule.