

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 , $v(M) = \text{True}$ implies $v(C) = \text{True}$.

Abstract DPLL Rules

UnitProp :

$$M \parallel F, C \vee I \quad \Longrightarrow \quad M I \parallel F, C \vee I \quad \mathbf{if} \quad \left\{ \begin{array}{l} M \models \neg C \\ I \text{ is undefined in } M \end{array} \right.$$

Abstract DPLL Rules

UnitProp :

$$M \parallel F, C \vee I \quad \Longrightarrow \quad M I \parallel F, C \vee I \quad \mathbf{if} \quad \left\{ \begin{array}{l} M \models \neg C \\ I \text{ is undefined in } M \end{array} \right.$$

PureLiteral :

$$M \parallel F \quad \Longrightarrow \quad M I \parallel F \quad \mathbf{if} \quad \left\{ \begin{array}{l} I \text{ occurs in some clause of } F \\ \neg I \text{ occurs in no clause of } F \\ I \text{ is undefined in } M \end{array} \right.$$

Abstract DPLL Rules

UnitProp :

$$M \parallel F, C \vee l \quad \Longrightarrow \quad M l \parallel F, C \vee l \quad \mathbf{if} \quad \left\{ \begin{array}{l} M \models \neg C \\ l \text{ is undefined in } M \end{array} \right.$$

PureLiteral :

$$M \parallel F \quad \Longrightarrow \quad M l \parallel F \quad \mathbf{if} \quad \left\{ \begin{array}{l} l \text{ occurs in some clause of } F \\ \neg l \text{ occurs in no clause of } F \\ l \text{ is undefined in } M \end{array} \right.$$

Decide :

$$M \parallel F \quad \Longrightarrow \quad M l^d \parallel F \quad \mathbf{if} \quad \left\{ \begin{array}{l} l \text{ or } \neg l \text{ occurs in a clause of } F \\ l \text{ is undefined in } M \end{array} \right.$$

Abstract DPLL Rules

UnitProp :

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

PureLiteral :

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

Decide :

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

Backtrack :

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

Abstract DPLL Rules

UnitProp :

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

PureLiteral :

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

Decide :

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

Backtrack :

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

Fail :

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

Example

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

Example

$$\begin{array}{l} \emptyset \parallel 1\vee\bar{2}, \bar{1}\vee\bar{2}, 2\vee 3, \bar{3}\vee 2, 1\vee 4 \\ 4 \parallel 1\vee\bar{2}, \bar{1}\vee\bar{2}, 2\vee 3, \bar{3}\vee 2, 1\vee 4 \end{array} \implies (\text{PureLiteral})$$

Example

\emptyset	\parallel	$1\vee\bar{2}$,	$\bar{1}\vee\bar{2}$,	$2\vee 3$,	$\bar{3}\vee 2$,	$1\vee 4$	\implies	(PureLiteral)
4	\parallel	$1\vee\bar{2}$,	$\bar{1}\vee\bar{2}$,	$2\vee 3$,	$\bar{3}\vee 2$,	$1\vee 4$	\implies	(Decide)
4 1^d	\parallel	$1\vee\bar{2}$,	$\bar{1}\vee\bar{2}$,	$2\vee 3$,	$\bar{3}\vee 2$,	$1\vee 4$		

Example

\emptyset		$1\sqrt{2}$,	$\bar{1}\sqrt{2}$,	$2\sqrt{3}$,	$\bar{3}\sqrt{2}$,	$1\sqrt{4}$	\implies	(PureLiteral)
4		$1\sqrt{2}$,	$\bar{1}\sqrt{2}$,	$2\sqrt{3}$,	$\bar{3}\sqrt{2}$,	$1\sqrt{4}$	\implies	(Decide)
4 1^d		$1\sqrt{2}$,	$\bar{1}\sqrt{2}$,	$2\sqrt{3}$,	$\bar{3}\sqrt{2}$,	$1\sqrt{4}$	\implies	(UnitProp)
4 $1^d \bar{2}$		$1\sqrt{2}$,	$\bar{1}\sqrt{2}$,	$2\sqrt{3}$,	$\bar{3}\sqrt{2}$,	$1\sqrt{4}$		

Example

\emptyset		$1\vee\bar{2}$,	$\bar{1}\vee\bar{2}$,	$2\vee 3$,	$\bar{3}\vee 2$,	$1\vee 4$	\implies	(PureLiteral)
4		$1\vee\bar{2}$,	$\bar{1}\vee\bar{2}$,	$2\vee 3$,	$\bar{3}\vee 2$,	$1\vee 4$	\implies	(Decide)
4 1^d		$1\vee\bar{2}$,	$\bar{1}\vee\bar{2}$,	$2\vee 3$,	$\bar{3}\vee 2$,	$1\vee 4$	\implies	(UnitProp)
4 $1^d \bar{2}$		$1\vee\bar{2}$,	$\bar{1}\vee\bar{2}$,	$2\vee 3$,	$\bar{3}\vee 2$,	$1\vee 4$	\implies	(UnitProp)
4 $1^d \bar{2} 3$		$1\vee\bar{2}$,	$\bar{1}\vee\bar{2}$,	$2\vee 3$,	$\bar{3}\vee 2$,	$1\vee 4$		

Example

\emptyset		$1\vee\bar{2}$,	$\bar{1}\vee\bar{2}$,	$2\vee 3$,	$\bar{3}\vee 2$,	$1\vee 4$	\implies	(PureLiteral)
4		$1\vee\bar{2}$,	$\bar{1}\vee\bar{2}$,	$2\vee 3$,	$\bar{3}\vee 2$,	$1\vee 4$	\implies	(Decide)
4 1^d		$1\vee\bar{2}$,	$\bar{1}\vee\bar{2}$,	$2\vee 3$,	$\bar{3}\vee 2$,	$1\vee 4$	\implies	(UnitProp)
4 $1^d \bar{2}$		$1\vee\bar{2}$,	$\bar{1}\vee\bar{2}$,	$2\vee 3$,	$\bar{3}\vee 2$,	$1\vee 4$	\implies	(UnitProp)
4 $1^d \bar{2} 3$		$1\vee\bar{2}$,	$\bar{1}\vee\bar{2}$,	$2\vee 3$,	$\bar{3}\vee 2$,	$1\vee 4$	\implies	(Backtrack)
4 $\bar{1}$		$1\vee\bar{2}$,	$\bar{1}\vee\bar{2}$,	$2\vee 3$,	$\bar{3}\vee 2$,	$1\vee 4$		

Example

\emptyset		$1\bar{v}2,$	$\bar{1}v\bar{2},$	$2v3,$	$\bar{3}v2,$	$1v4$	\implies	(PureLiteral)
4		$1\bar{v}2,$	$\bar{1}v\bar{2},$	$2v3,$	$\bar{3}v2,$	$1v4$	\implies	(Decide)
4 1^d		$1\bar{v}2,$	$\bar{1}v\bar{2},$	$2v3,$	$\bar{3}v2,$	$1v4$	\implies	(UnitProp)
4 $1^d \bar{2}$		$1\bar{v}2,$	$\bar{1}v\bar{2},$	$2v3,$	$\bar{3}v2,$	$1v4$	\implies	(UnitProp)
4 $1^d \bar{2} 3$		$1\bar{v}2,$	$\bar{1}v\bar{2},$	$2v3,$	$\bar{3}v2,$	$1v4$	\implies	(Backtrack)
4 $\bar{1}$		$1\bar{v}2,$	$\bar{1}v\bar{2},$	$2v3,$	$\bar{3}v2,$	$1v4$	\implies	(UnitProp)
4 $\bar{1} \bar{2} \bar{3}$		$1\bar{v}2,$	$\bar{1}v\bar{2},$	$2v3,$	$\bar{3}v2,$	$1v4$	\implies	(UnitProp)

Example

\emptyset		$1\bar{v}2$,	$\bar{1}v\bar{2}$,	$2v3$,	$\bar{3}v2$,	$1v4$	\implies	(PureLiteral)
4		$1\bar{v}2$,	$\bar{1}v\bar{2}$,	$2v3$,	$\bar{3}v2$,	$1v4$	\implies	(Decide)
4 1 ^d		$1\bar{v}2$,	$\bar{1}v\bar{2}$,	$2v3$,	$\bar{3}v2$,	$1v4$	\implies	(UnitProp)
4 1 ^d $\bar{2}$		$1\bar{v}2$,	$\bar{1}v\bar{2}$,	$2v3$,	$\bar{3}v2$,	$1v4$	\implies	(UnitProp)
4 1 ^d $\bar{2}$ 3		$1\bar{v}2$,	$\bar{1}v\bar{2}$,	$2v3$,	$\bar{3}v2$,	$1v4$	\implies	(Backtrack)
4 $\bar{1}$		$1\bar{v}2$,	$\bar{1}v\bar{2}$,	$2v3$,	$\bar{3}v2$,	$1v4$	\implies	(UnitProp)
4 $\bar{1}$ $\bar{2}$ $\bar{3}$		$1\bar{v}2$,	$\bar{1}v\bar{2}$,	$2v3$,	$\bar{3}v2$,	$1v4$	\implies	(Fail)
								<i>fail</i>

Example

\emptyset		$1\bar{v}2$,	$\bar{1}v\bar{2}$,	$2v3$,	$\bar{3}v2$,	$1v4$	\implies	(PureLiteral)
4		$1\bar{v}2$,	$\bar{1}v\bar{2}$,	$2v3$,	$\bar{3}v2$,	$1v4$	\implies	(Decide)
4 1^d		$1\bar{v}2$,	$\bar{1}v\bar{2}$,	$2v3$,	$\bar{3}v2$,	$1v4$	\implies	(UnitProp)
4 $1^d \bar{2}$		$1\bar{v}2$,	$\bar{1}v\bar{2}$,	$2v3$,	$\bar{3}v2$,	$1v4$	\implies	(UnitProp)
4 $1^d \bar{2} 3$		$1\bar{v}2$,	$\bar{1}v\bar{2}$,	$2v3$,	$\bar{3}v2$,	$1v4$	\implies	(Backtrack)
4 $\bar{1}$		$1\bar{v}2$,	$\bar{1}v\bar{2}$,	$2v3$,	$\bar{3}v2$,	$1v4$	\implies	(UnitProp)
4 $\bar{1} \bar{2} \bar{3}$		$1\bar{v}2$,	$\bar{1}v\bar{2}$,	$2v3$,	$\bar{3}v2$,	$1v4$	\implies	(Fail)
								<i>fail</i>

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:

Learn :

$$M \parallel F \quad \Longrightarrow \quad M \parallel F, C \quad \text{if} \quad \left\{ \begin{array}{l} \text{all atoms of } C \text{ occur in } F \\ F \models C \end{array} \right.$$

Backjump :

$$M \uparrow^d N \parallel F, C \quad \Longrightarrow \quad M \uparrow' \parallel F, C \quad \text{if} \quad \left\{ \begin{array}{l} M \uparrow^d N \models \neg C, \text{ and there is} \\ \text{some clause } C' \vee I' \text{ such that :} \\ F, C \models C' \vee I' \text{ and } M \models \neg C', \\ I' \text{ is undefined in } M, \text{ and} \\ I' \text{ or } \neg I' \text{ occurs in } F \text{ or in } M \uparrow^d N \end{array} \right.$$

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.

Example

$\emptyset \parallel \bar{1}\vee 2, 3\vee 4, \bar{5}\vee \bar{6}, \bar{2}\vee \bar{5}\vee 6$

Example

$$\begin{array}{l} \emptyset \parallel \bar{1}\vee 2, 3\vee 4, \bar{5}\vee \bar{6}, \bar{2}\vee \bar{5}\vee 6 \\ 1^d \parallel \bar{1}\vee 2, 3\vee 4, \bar{5}\vee \bar{6}, \bar{2}\vee \bar{5}\vee 6 \end{array} \implies \text{(Decide)}$$

Example

\emptyset	\parallel	$\bar{1}\vee 2$, $3\vee 4$, $\bar{5}\vee \bar{6}$, $\bar{2}\vee \bar{5}\vee 6$	\implies	(Decide)
1^d	\parallel	$\bar{1}\vee 2$, $3\vee 4$, $\bar{5}\vee \bar{6}$, $\bar{2}\vee \bar{5}\vee 6$	\implies	(UnitProp)
$1^d 2$	\parallel	$\bar{1}\vee 2$, $3\vee 4$, $\bar{5}\vee \bar{6}$, $\bar{2}\vee \bar{5}\vee 6$		

Example

\emptyset		$\bar{1}\vee 2$,	$3\vee 4$,	$\bar{5}\vee \bar{6}$,	$\bar{2}\vee \bar{5}\vee 6$	\implies	(Decide)
1^d		$\bar{1}\vee 2$,	$3\vee 4$,	$\bar{5}\vee \bar{6}$,	$\bar{2}\vee \bar{5}\vee 6$	\implies	(UnitProp)
$1^d 2$		$\bar{1}\vee 2$,	$3\vee 4$,	$\bar{5}\vee \bar{6}$,	$\bar{2}\vee \bar{5}\vee 6$	\implies	(Decide)
$1^d 2 3^d$		$\bar{1}\vee 2$,	$3\vee 4$,	$\bar{5}\vee \bar{6}$,	$\bar{2}\vee \bar{5}\vee 6$		

Example

\emptyset		$\bar{1}\vee 2$,	$3\vee 4$,	$\bar{5}\vee \bar{6}$,	$\bar{2}\vee \bar{5}\vee 6$	\implies	(Decide)
1^d		$\bar{1}\vee 2$,	$3\vee 4$,	$\bar{5}\vee \bar{6}$,	$\bar{2}\vee \bar{5}\vee 6$	\implies	(UnitProp)
$1^d 2$		$\bar{1}\vee 2$,	$3\vee 4$,	$\bar{5}\vee \bar{6}$,	$\bar{2}\vee \bar{5}\vee 6$	\implies	(Decide)
$1^d 2 3^d$		$\bar{1}\vee 2$,	$3\vee 4$,	$\bar{5}\vee \bar{6}$,	$\bar{2}\vee \bar{5}\vee 6$	\implies	(Decide)
$1^d 2 3^d 5^d$		$\bar{1}\vee 2$,	$3\vee 4$,	$\bar{5}\vee \bar{6}$,	$\bar{2}\vee \bar{5}\vee 6$		

Example

\emptyset		$\bar{1}V2$,	$3V4$,	$\bar{5}V\bar{6}$,	$\bar{2}V\bar{5}V6$	\implies	(Decide)
1^d		$\bar{1}V2$,	$3V4$,	$\bar{5}V\bar{6}$,	$\bar{2}V\bar{5}V6$	\implies	(UnitProp)
$1^d 2$		$\bar{1}V2$,	$3V4$,	$\bar{5}V\bar{6}$,	$\bar{2}V\bar{5}V6$	\implies	(Decide)
$1^d 2 3^d$		$\bar{1}V2$,	$3V4$,	$\bar{5}V\bar{6}$,	$\bar{2}V\bar{5}V6$	\implies	(Decide)
$1^d 2 3^d 5^d$		$\bar{1}V2$,	$3V4$,	$\bar{5}V\bar{6}$,	$\bar{2}V\bar{5}V6$	\implies	(UnitProp)
$1^d 2 3^d 5^d \bar{6}$		$\bar{1}V2$,	$3V4$,	$\bar{5}V\bar{6}$,	$\bar{2}V\bar{5}V6$		

Example

\emptyset		$\bar{1}V2$, $3V4$, $\bar{5}V\bar{6}$, $\bar{2}V\bar{5}V6$	\implies	(Decide)
1^d		$\bar{1}V2$, $3V4$, $\bar{5}V\bar{6}$, $\bar{2}V\bar{5}V6$	\implies	(UnitProp)
$1^d 2$		$\bar{1}V2$, $3V4$, $\bar{5}V\bar{6}$, $\bar{2}V\bar{5}V6$	\implies	(Decide)
$1^d 2 3^d$		$\bar{1}V2$, $3V4$, $\bar{5}V\bar{6}$, $\bar{2}V\bar{5}V6$	\implies	(Decide)
$1^d 2 3^d 5^d$		$\bar{1}V2$, $3V4$, $\bar{5}V\bar{6}$, $\bar{2}V\bar{5}V6$	\implies	(UnitProp)
$1^d 2 3^d 5^d \bar{6}$		$\bar{1}V2$, $3V4$, $\bar{5}V\bar{6}$, $\bar{2}V\bar{5}V6$	\implies	(Learn)
$1^d 2 3^d 5^d \bar{6}$		$\bar{1}V2$, $3V4$, $\bar{5}V\bar{6}$, $\bar{2}V\bar{5}V6$, $\bar{2}V\bar{5}$		

Example

\emptyset		$\bar{1}v2$, $3v4$, $\bar{5}v\bar{6}$, $\bar{2}v\bar{5}v6$	\implies	(Decide)
1^d		$\bar{1}v2$, $3v4$, $\bar{5}v\bar{6}$, $\bar{2}v\bar{5}v6$	\implies	(UnitProp)
$1^d 2$		$\bar{1}v2$, $3v4$, $\bar{5}v\bar{6}$, $\bar{2}v\bar{5}v6$	\implies	(Decide)
$1^d 2 3^d$		$\bar{1}v2$, $3v4$, $\bar{5}v\bar{6}$, $\bar{2}v\bar{5}v6$	\implies	(Decide)
$1^d 2 3^d 5^d$		$\bar{1}v2$, $3v4$, $\bar{5}v\bar{6}$, $\bar{2}v\bar{5}v6$	\implies	(UnitProp)
$1^d 2 3^d 5^d \bar{6}$		$\bar{1}v2$, $3v4$, $\bar{5}v\bar{6}$, $\bar{2}v\bar{5}v6$	\implies	(Learn)
$1^d 2 3^d 5^d \bar{6}$		$\bar{1}v2$, $3v4$, $\bar{5}v\bar{6}$, $\bar{2}v\bar{5}v6$, $\bar{2}v\bar{5}$		
		\implies (Backjump)		
$1^d 2 \bar{5}$		$\bar{1}v2$, $3v4$, $\bar{5}v\bar{6}$, $\bar{2}v\bar{5}v6$, $\bar{2}v\bar{5}$		

Example

\emptyset		$\bar{1}v2, 3v4, \bar{5}v\bar{6}, \bar{2}v\bar{5}v6$	\implies	(Decide)
1^d		$\bar{1}v2, 3v4, \bar{5}v\bar{6}, \bar{2}v\bar{5}v6$	\implies	(UnitProp)
$1^d 2$		$\bar{1}v2, 3v4, \bar{5}v\bar{6}, \bar{2}v\bar{5}v6$	\implies	(Decide)
$1^d 2 3^d$		$\bar{1}v2, 3v4, \bar{5}v\bar{6}, \bar{2}v\bar{5}v6$	\implies	(Decide)
$1^d 2 3^d 5^d$		$\bar{1}v2, 3v4, \bar{5}v\bar{6}, \bar{2}v\bar{5}v6$	\implies	(UnitProp)
$1^d 2 3^d 5^d \bar{6}$		$\bar{1}v2, 3v4, \bar{5}v\bar{6}, \bar{2}v\bar{5}v6$	\implies	(Learn)
$1^d 2 3^d 5^d \bar{6}$		$\bar{1}v2, 3v4, \bar{5}v\bar{6}, \bar{2}v\bar{5}v6, \bar{2}v\bar{5}$		
		\implies (Backjump)		
$1^d 2 \bar{5}$		$\bar{1}v2, 3v4, \bar{5}v\bar{6}, \bar{2}v\bar{5}v6, \bar{2}v\bar{5}$		
		\implies (Decide)		
$1^d 2 \bar{5} 3^d$		$\bar{1}v2, 3v4, \bar{5}v\bar{6}, \bar{2}v\bar{5}v6, \bar{2}v\bar{5}$		

Example

\emptyset		$\bar{1}v2, 3v4, \bar{5}v\bar{6}, \bar{2}v\bar{5}v6$	\implies	(Decide)
1^d		$\bar{1}v2, 3v4, \bar{5}v\bar{6}, \bar{2}v\bar{5}v6$	\implies	(UnitProp)
$1^d 2$		$\bar{1}v2, 3v4, \bar{5}v\bar{6}, \bar{2}v\bar{5}v6$	\implies	(Decide)
$1^d 2 3^d$		$\bar{1}v2, 3v4, \bar{5}v\bar{6}, \bar{2}v\bar{5}v6$	\implies	(Decide)
$1^d 2 3^d 5^d$		$\bar{1}v2, 3v4, \bar{5}v\bar{6}, \bar{2}v\bar{5}v6$	\implies	(UnitProp)
$1^d 2 3^d 5^d \bar{6}$		$\bar{1}v2, 3v4, \bar{5}v\bar{6}, \bar{2}v\bar{5}v6$	\implies	(Learn)
$1^d 2 3^d 5^d \bar{6}$		$\bar{1}v2, 3v4, \bar{5}v\bar{6}, \bar{2}v\bar{5}v6, \bar{2}v\bar{5}$		
		\implies	(Backjump)	
$1^d 2 \bar{5}$		$\bar{1}v2, 3v4, \bar{5}v\bar{6}, \bar{2}v\bar{5}v6, \bar{2}v\bar{5}$		
		\implies	(Decide)	
$1^d 2 \bar{5} 3^d$		$\bar{1}v2, 3v4, \bar{5}v\bar{6}, \bar{2}v\bar{5}v6, \bar{2}v\bar{5}$		

Result: *Satisfiable*

Abstract DPLL Rules

Two final rules also have to do with learning:

- ▶ If too many clauses are learned, performance suffers. It is useful to *forget* some clauses (typically those that have not participated in an application of **UnitProp** for a while).
- ▶ If we are stuck, we can *restart* by throwing away M . Since we have learned clauses, this means our efforts were not entirely wasted. Randomly restarting can improve performance dramatically.

Forget :

$$M \parallel F, C \implies M \parallel F \quad \text{if } \{ F \models C$$

Restart :

$$M \parallel F \implies \emptyset \parallel F$$

Decision Heuristics

The rules do not give any strategy for *how* to pick a variable when applying *Decide*.

In practice, this is critical for performance.

There are many heuristics, but the most successful currently use very cheap heuristics to try to prefer variables that are frequently involved in conflicts.

Boolean Constraint Propagation

The most expensive part of a SAT solver is the part that checks for and applies instances of the **UnitProp** rule.

A key insight that can be used to speed this up is that as long as a clause has at least two unassigned literals, it cannot participate in an application of **UnitProp**.

For every clause, we assign two of its unassigned literals as the *watched* literals.

Every time a literal is assigned, only those clauses in which it is watched need to be checked for a possible triggering of the **UnitProp** rule.

For those clauses that are inspected, if **UnitProp** is not triggered, a new unassigned literal is chosen to be watched.

Other Considerations

Modern SAT solvers have a number of other tricks to speed things up:

- ▶ Highly tuned code
- ▶ Optimization for cache performance
- ▶ Preprocessing and clever CNF encodings
- ▶ Automatic tuning of program parameters