Chapter 3

Watched Literals and Algorithm 5

The adoption of clause learning and non-chronological backtracking can greatly improve the performance of satisfiability solvers. One possible way to improve the performance further would be to reduce the time spent by the solvers while backtracking. One way to reduce time while backtracking is to use an alternative approach for detecting unit literals, satisfied clauses and falsified clauses (with respect to the partial interpretation) known as "watched literals" [18]. The watched literals technique improves efficiency while backtracking and may result in improving overall efficiency.

In watched literals we maintain two "watched literal" pointers with each clause. At all times, the two watched literal pointers point to distinct literals in the clause, if there are at least two literals. If there is only one literal in the clause, then the first pointer points to it while the second pointer is $NULL$. (We assume there are no empty clauses in the theory.) From then on, we maintain the following four properties. (1) A clause is satisfied iff its first watched literal is satisfied. (2) A clause is falsified iff its first watched literal is falsified and the second watched literal is either falsified or $NULL$. (3) A clause is unit iff it has one of its watched literals falsified or $NULL$ and the other unassigned. (4) A clause is unassigned and not unit iff both its watched literals are unassigned.

So when a clause becomes satisfied, the first watched literal pointer is set to point to the satisfied literal in the clause. (And if it happens that the second watched literal is the one being satisfied, we swap the watched literal pointers.) If the clause is not yet satisfied and one of the watched literals is becoming false, we need to do some watched literal maintenance work. Suppose the watched literal other than the currently falsified watched literal already points to a falsified literal or is $NULL$. Then we have a falsified clause with respect to the current partial interpretation, and nothing need be done. Otherwise, if the watched literal other than the currently falsified watched literal is unassigned then we need to update the current falsified watched literal to point to an unassigned literal in the clause different than the other watched literal, if one is available. If none is available, then we have a unit clause with unit literal the other watched literal, and nothing need be done.

In Algorithm 5, we will present the watched literals version of Algorithm 4. We still maintain global data structures for representing the current partial interpretation and keeping track of our progress toward satisfying the theory, but now using watched literals for some of the work.

Each literal L has again two associated fields, $L.value$ and $L.level$, and they work the same as in Algorithm 4.

A clause C now has two associated fields: $C.W1$, and $C.W2$. If the clause C consists of only one literal then $C.W1$ points to the only literal in the clause and $C.W2$ is $NULL$ (null pointer). Otherwise, $C.W1$ and $C.W2$ always point to distinct literals in the clause C. The watched literal $C.W1$ will be used to keep track of whether C is currently satisfied; C is satisfied iff $C.W1.value = t$. Moreover, if C is satisfied, we can now obtain the level at which it became satisfied by looking at $C.W1.level$. This helps a lot while backtracking, as we no longer have to un-set any *sat* flag and *level* field associated with the clause, since the $W1's$ *value* and *level* fields take care of this. This improves efficiency while backtracking and may improve efficiency overall. We also no longer maintain a *count* field for each clause.

Algorithm 5

```
Solve()
 for all literals $L$ do
  $L.value \leftarrow u$
  $L.level \leftarrow -1$
 for all clauses $C$ do
  $C.W1 \leftarrow$ points to the first literal in clause $C$
  $C.W2 \leftarrow$ points to the second literal in clause $C$, otherwise $NULL$
 DPLL()
```

```
checkSat()
 if there is a clause $C$ s.t. $C.W1.value = f$ and
 $(C.W2.value = f$ or $C.W2.value = NULL)$ then
  return $f$
 else if for every clause $C$, $C.W1.value = t$ then
  return $t$
 else
  return $u$
```

```
findUnitLiteral()
 for all clauses $C$ do
  if $(C.W1.value = u)$ then
   if $(C.W2.value = NULL$ or $C.W2.value = f)$ then
    return $C.W1$
  if $(C.W2.value = u)$ then
   if $(C.W1.value = f)$ then
    return $C.W2$
 return $NULL$
```

```
pickLiteral()
 return (a literal $L$ s.t. $L.value = u$ and for some clause $C$, $L \in C$ s.t. $C.W1.value = u$)
```

```
propagate($L, level$)
 $P \leftarrow \{L \mid L.value = t\}$
 for all literals $L'$ s.t. $P \cup \{L\} \models L'$ do
  if ($L'.value = u$) then
   $L'.value \leftarrow t$
   $L'.level \leftarrow level$
   for all clauses $C$ s.t. $L' \in C$ do
    if ($C.W1.value \neq t$) then
     if ($C.W2 \neq L'$) then
      $C.W1 \leftarrow L'$
     else
      $C.W2 \leftarrow C.W1$
      $C.W1 \leftarrow L'$
 for all literals $L'$ s.t. $P \cup \{L\} \Dashv L'$ do
  if ($L'.value = u$) then
   $L'.value \leftarrow f$
   $L'.level \leftarrow level$
   for all clauses $C$ with $L' \in C$ do
    if ($C.W1.value \neq t$) then
     if ($C.W1 = L'$ or $C.W2 = L'$) then
      swapPointer($C$)
```

```
swapPointer($C$)
 if ($C.W2.value = u$) then
  if there exists $L \in C$ s.t. $L \neq C.W2$ and $L.value = u$ then
   $C.W1 \leftarrow L$
 if ($C.W1.value = u$) then
  if there exists $L \in C$ s.t. $L \neq C.W1$ and $L.value = u$ then
   $C.W2 \leftarrow L$
```

```
backtrack($level$)
 for all literals $L$ do
  if ($L.level > level$) then
   $L.value \leftarrow u$
   $L.level \leftarrow -1$
```

As before, **Solve()** starts with initializing the data structures and makes a call **DPLL()**. Please note that **DPLL()** is the same as in algorithm 4. Again, the call **DPLL()** will return the partial interpretation based on the properties of **analyzeConflict**() and **findBacktrackingLevel**(*learnedClause*); otherwise *false* is returned.

Most auxiliary functions perform the same function as in Algorithm 3 but a bit differently as we see below:

checkSat(): This function checks to see if $P \models T$, $P \Dashv T$ or otherwise. If $P \models T$ it returns t. If $P \Dashv T$ it returns f. Otherwise it returns u. This is now accomplished by looking at the watched literals.

findUnitLiteral(): This function returns a unit literal w.r.t. P if one exists. Otherwise, it returns $NULL$. Again, this is done by looking at watched literals.

pickLiteral(): This function returns an unassigned random literal from a clause not satisfied with respect to the current partial interpretation. Here too watched literals are used.

propagate($L, level$**):** This function satisfies all literals L' s.t. $P \cup \{L\} \models L'$ and $P \not\models L'$. The function also satisfies all not yet satisfied clauses C containing such L' with respect to P, and then makes sure the newly satisfied literal is $C.W1$. This function falsifies unassigned literals L' s.t. $P \cup \{L\} \Dashv L'$ and for all not yet satisfied clauses C having L' as one of their watched literals calls function **swapPointer**(C)**.**

swapPointer(C**):** If one of the watched literals in clause C is unassigned and other is falsified then the function tries to find an unassigned literal replacement for the falsified literal other than the already unassigned watched literal.

backtrack($level$**):** This function restores the state just before the decision that began level $level + 1$. Notice that, on comparison, the **backtrack()** function in Algorithm 5 does almost no work compared to the one in Algorithm 4.

Let's look at the Example 1 used in Algorithm 4 now solved using Algorithm 5!

As before, the current partial interpretation $P = \emptyset$. **Solve()** initializes the data structures and make the first call **DPLL()**.

$P = \emptyset$

$Clause1\ \{1 = 1\}$	U	$W1(1 = 1)$	$W2(NULL)$
$Clause2\ \{1 = 0,\ 2 = 0,\ 3 = 1\}$	U	$W1(1 = 0)$	$W2(2 = 0)$
$Clause3\ \{1 = 0,\ 2 = 1,\ 3 = 0\}$	U	$W1(1 = 0)$	$W2(2 = 1)$
$Clause4\ \{1 = 0,\ 2 = 2,\ 3 \neq 0\}$	U	$W1(1 = 0)$	$W2(2 = 2)$

$W1$ and $W2$ after the symbol U represent the watched literals in a clause.

$Clause1$ is unit as $Clause1.W1.value = u$ and $Clause1.W2.value = NULL$, so $1 = 1$ is a unit literal. So literal $1 = 1$ is propagated at level 0.

$P = \{1 = 1_0\}$

$Clause1\ \{\mathit{1} = \mathit{1}\}$	S	$W1(\mathit{1} = \mathit{1})$	$W2(NULL)$
$Clause2\ \{$~~$1 = 0$~~$,\ 2 = 0,\ 3 = 1\}$	U	$W1(3 = 1)$	$W2(2 = 0)$
$Clause3\ \{$~~$1 = 0$~~$,\ 2 = 1,\ 3 = 0\}$	U	$W1(3 = 0)$	$W2(2 = 1)$
$Clause4\ \{$~~$1 = 0$~~$,\ 2 = 2,\ 3 \neq 0\}$	U	$W1(3 \neq 0)$	$W2(2 = 2)$

Literal $1 = 0$ is falsified. $Clause2$, $Clause3$ and $Clause4$ are not satisfied and have literal $1 = 0$ as their first watched literal, so watched literal maintenance is done and the first watched literal in these three clauses is replaced by an unassigned literal. Thus $Clause2.W1$ becomes $3 = 1$, $Clause3.W1$ becomes $3 = 0$ and $Clause4.W1$ becomes $3 \neq 0$.

Now T is neither satisfied nor falsified and there are no more unit literals, so we pick a literal. Assume the picked literal is $2 = 0$ from $Clause2$. The $level$ is incremented to 1 and literal $2 = 0$ is propagated.

$P = \{1 = 1_0,\ 2 = 0_1\}$

$Clause1\ \{\mathit{1} = \mathit{1}\}$	S	$W1(\mathit{1} = \mathit{1})$	$W2(NULL)$
$Clause2\ \{$~~$1 = 0$~~$,\ 2 = \mathit{0},\ 3 = 1\}$	S	$W1(2 = \mathit{0})$	$W2(3 = 1)$
$Clause3\ \{$~~$1 = 0$~~, ~~$2 = 1$~~$,\ 3 = 0\}$	U	$W1(3 = 0)$	$W2($~~$2 = 1$~~$)$
$Clause4\ \{$~~$1 = 0$~~, ~~$2 = 2$~~$,\ 3 \neq 0\}$	U	$W1(3 \neq 0)$	$W2($~~$2 = 2$~~$)$

Now $Clause3$ is unit as $Clause3.W1.value = u$ and $Clause3.W2.value = f$. So we must satisfy unit literal $3 = 0$ in $Clause3$. So literal $3 = 0$ is propagated at level 0.

$P = \{1 = 1_0,\ 2 = 0_1,\ 3 = 0_1\}$

$Clause1$	{$1 = 1$}	S	$W1$($1 = 1$)	$W2(NULL)$
$Clause2$	{~~$1 = 0$~~, $2 = 0$, ~~$3 = 1$~~}	S	$W1$($2 = 0$)	$W2$(~~$3 = 1$~~)
$Clause3$	{~~$1 = 0$~~, ~~$2 = 1$~~, $3 = 0$}	S	$W1$($3 = 0$)	$W2$(~~$2 = 1$~~)
$Clause4$	{~~$1 = 0$~~, ~~$2 = 2$~~, ~~$3 \neq 0$~~}	F	$W1$(~~$3 \neq 0$~~)	$W2$(~~$2 = 2$~~)

Here T is falsified because $Clause4.W1.value = f$ and $Clause4.W2.value = f$. As current level is not equal to 0, $Clause5$ is learned and added to the theory and we backtrack to level 0 based on learned $Clause5$.

$P = \{1 = 1_0\}$

$Clause1$	{$1 = 1$}	S	$W1$($1 = 1$)	$W2(NULL)$
$Clause2$	{~~$1 = 0$~~, $2 = 0,\ 3 = 1$}	U	$W1(2 = 0)$	$W2(3 = 1)$
$Clause3$	{~~$1 = 0$~~, $2 = 1,\ 3 = 0$}	U	$W1(3 = 0)$	$W2(2 = 1)$
$Clause4$	{~~$1 = 0$~~, $2 = 2,\ 3 \neq 0$}	U	$W1(3 \neq 0)$	$W2(2 = 2)$
$Clause5$	{~~$1 = 0$~~, $2 \neq 0$}	U	$W1(2 \neq 0)$	$W2$(~~$1 = 0$~~)

After backtracking to level 0, $Clause5$ is unit because $Clause5.W1.value = u$ and $Clause5.W2.value = f$. So we must satisfy unit literal $2 \neq 0$ in $Clause5$. Thus, literal $2 \neq 0$ is propagated at level 0.

$P = \{1 = 1_0,\ 2 \neq 0_0\}$

$Clause1$	{$1 = 1$}	S	$W1$($1 = 1$)	$W2(NULL)$
$Clause2$	{~~$1 = 0$~~, ~~$2 = 0$~~, $3 = 1$}	U	$W1$(~~$2 = 0$~~)	$W2(3 = 1)$
$Clause3$	{~~$1 = 0$~~, $2 = 1,\ 3 = 0$}	U	$W1(3 = 0)$	$W2(2 = 1)$
$Clause4$	{~~$1 = 0$~~, $2 = 2,\ 3 \neq 0$}	U	$W1(3 \neq 0)$	$W2(2 = 2)$
$Clause5$	{~~$1 = 0$~~, $2 \neq 0$}	S	$W1$($2 \neq 0$)	$W2$(~~$1 = 0$~~)

Now $Clause2$ is unit as $Clause2.W1.value = f$ and $Clause2.W2.value = u$. So we must satisfy unit literal $3 = 1$ in $Clause2$. So literal $3 = 1$ is propagated at level 0.