问题变量的赋值可以定义为函数$\nu : X \to \{0, u, 1\}$，其中 u 表示当变量未在{0, 1}中赋值时使用的未定义值。给定一个赋值ν，如果所有变量都在{0, 1}中赋值，则ν被称为一个完整的赋值。否则就是部分赋值。赋值用于计算文字、子句和完整的 CNF 公式的值，分别为l^ν, ω^ν and φ^ν

赋值函数 ν 也将被视为一组元组(x_i, v_i)，其中 vi ∈{0, 1}。将元组(x_i, v_i)添加到 ν 对应于将 vi 分配给 xi，使得$\nu(x_i) = v_i$。 从 ν 中移除元组(x_i, v_i)，其中$\nu(x_i) \neq u$，对应于将 u 分配给 xi。

子句的特征为不满意、满意、单元或未解决。如果一个子句的所有文字都被赋值为 0，则子句不满足。如果一个子句的至少一个文字被赋值为 1，则该子句满足。如果除一个之外的所有文字都被赋值 0，并且其余文字未被赋值，则子句是单元的。最后，如果一个子句既不是不满足的，也不是满足的，也不是单位的，那么它就是未解决的。

SAT 求解器中的一个关键过程是单元子句规则 [DP60]：如果一个子句是单元子句，那么它唯一未分配的文字必须赋值为 1 才能满足该子句。单元传播在每个分支步骤之后（以及在预处理期间）应用，并用于识别必须分配特定布尔值的变量。 如果识别出不满足的子句，则声明冲突条件，并且算法回溯。

在 CDCL SAT 求解器中，每个变量 xi 由许多属性表征，包括值、前因和决策水平，分别表示为$\nu(v_i) \in \{0, u, 1\}$, $\alpha(x_i) \in \varphi \cup \{\text{NIL}\}$, and $\delta(x_i) \in \{-1, 0, 1, \ldots, |X|\}$。

由于应用单元子句规则而被赋值的变量xi 被称为隐含的。用于蕴含变量 xi 的单元子句 ω 被称为 xi 的先行词，$\alpha(x_i) = \omega$。对于决策变量或未分配的变量，前因是 NIL。未赋值变量 xi 的决策水平为 −1，$\delta(x_i) = -1$。

与决策分配相关联的变量 xi 具有$\alpha(x_i) = \text{NIL and } \delta(x_i) > 0.$。

隐含文字的决策级别是单元子句中隐含文字的最高决策级别，或者在子句是单元的情况下为 0。符号 xi = v @ d 用于表示 ν(xi)=v 和 δ(xi)=d。蕴涵图中的顶点由所有分配的变量和一个特殊节点 κ 定义，Vl ⊆ X ∪{ κ }。如果单元传播产生一个不满足的子句 ωj，那么一个特殊的顶点 κ 被用来表示不满足的子句。在这种情况下，κ 的前件由 α(κ)= ωj 定义。

谓词 λ(z, ω) 取值 1 当且仅当ω中有文字 z，定义如下：

$$\lambda(z, \omega) = \begin{cases} 1 & \text{if } z \in \omega \vee \neg z \in \omega \\ 0 & \text{otherwise} \end{cases}$$

z 在 w 中且含 z 的文字取值为 0 则 v0(z,w)=1，v1 同理：

$$\nu_0(z,\omega) = \begin{cases} 1 & \text{if } \lambda(z,\omega) \wedge z \in \omega \wedge \nu(z) = 0 \\ 1 & \text{if } \lambda(z,\omega) \wedge \neg z \in \omega \wedge \nu(z) = 1 \\ 0 & \text{otherwise} \end{cases}$$

$$\nu_1(z,\omega) = \begin{cases} 1 & \text{if } \lambda(z,\omega) \wedge z \in \omega \wedge \nu(z) = 1 \\ 1 & \text{if } \lambda(z,\omega) \wedge \neg z \in \omega \wedge \nu(z) = 0 \\ 0 & \text{otherwise} \end{cases}$$

在 DPLL 式 SAT 求解器的执行过程中，分配的变量及其前因定义了一个有向无环图 I =(VI ,EI)，称为蕴涵图。在 I 中存在从 z1 到 z2 的边，当且仅当以下谓词取值 1：

$$\epsilon(z_1,z_2) = \begin{cases} 1 & \text{if } z_2 = \kappa \wedge \lambda(z_1, \alpha(\kappa)) \\ 1 & \text{if } z_2 \neq \kappa \wedge \alpha(z_2) = \omega \wedge \nu_0(z_1,\omega) \wedge \nu_1(z_2,\omega) \\ 0 & \text{otherwise} \end{cases}$$

因此，蕴涵图 I 的边集 EI 由下式给出：

$$E_I = \{(z_1,z_2) \mid \epsilon(z_1,z_2) = 1\}$$

算法 4.1 显示了 CDCL SAT 求解器的标准组织，它基本上遵循 DPLL 的组织。 对于 DPLL，主要区别在于每次识别冲突时调用 ConflictAnalysis 函数，以及在发生回溯时调用 Backtrack。 此外，回溯过程允许按时间顺序回溯。

Algorithm 4.1 Typical CDCL algorithm

```
CDCL(φ, ν)
1  if (UnitPropagation(φ, ν) == CONFLICT)
2     then return UNSAT
3  dl ← 0                                          ▷ Decision level
4  while (not AllVariablesAssigned(φ, ν))
5        do (x, v) = PickBranchingVariable(φ, ν)          ▷ Decide stage
6           dl ← dl + 1                  ▷ Increment decision level due to new decision
7           ν ← ν ∪ {(x, v)}
8           if (UnitPropagation(φ, ν) == CONFLICT)        ▷ Deduce stage
9              then β = ConflictAnalysis(φ, ν)            ▷ Diagnose stage
10                  if (β < 0)
11                     then return UNSAT
12                     else Backtrack(φ, ν, β)
13                          dl ← β          ▷ Decrement decision level due to backtracking
14 return SAT
```

除了主要的 CDCL 功能外，还使用了以下辅助功能：

UnitPropagation（单位传播）：如果识别出冲突，则返回 UNSAT。

PickBranchingVariable（选择分支变量）：选择要决策的变量和相应的值

ConflictAnalysis（冲突分析）：分析冲突并从冲突中学习新的子句并计算出要返回的决策层数

Backtrack（回溯）：回溯到 ConflictAnalysis 计算的决策层

AllVariablesAssigned（所有变量已分配）：测试是否已分配所有变量，在这种情况下算法终止，表明 CNF 公式是可满足的。

冲突分析

冲突中学习子句

冲突分析会学习一个或多个新的子句，并计算了回溯决策级别。

冲突分析过程访问处在当前最大决策级别的变量，识别每个变量的前因，但避免低于最大决策级别的文字的前因。重复此过程，直到访问最近的决策变量。

d 表示当前决策级别，xi 为决策变量，$\nu(x_i) = v$ 表示决策，wj 表示用单元传播标识的不可满足子句。两个子句 ω j 和 ω k，一个具有文字 x 而另一个具有文字¬x，则 $\omega_j \odot \omega_k$ 包含 ω j 和 ω k 的所有文字，x 和¬x 除外。

SAT 求解器中使用的子句学习过程可以通过一系列选择性解析操作来定义，每一步都会产生一个新的临时子句。首先，定义一个谓词，如果文字 l 存在于子句 w 中，l 的决策级别为 d，且 l 是被强制赋值的，则谓词取值 1：

$$\xi(\omega, l, d) = \begin{cases} 1 & \text{if } l \in \omega \wedge \delta(l) = d \wedge \alpha(l) \neq \text{NIL} \\ 0 & \text{otherwise} \end{cases}$$

$\omega_L^{d,i}$，表示经过 i 次解析操作得到的中间子句(临时子句)。

$$\omega_L^{d,i} = \begin{cases} \alpha(\kappa) & \text{if } i = 0 \\ \omega_L^{d,i-1} \odot \alpha(l) & \text{if } i \neq 0 \wedge \xi(\omega_L^{d,i-1}, l, d) = 1 \\ \omega_L^{d,i-1} & \text{if } i \neq 0 \wedge \forall_l \xi(\omega_L^{d,i-1}, l, d) = 0 \end{cases}$$

把 w(i-1)中每个在决策级别 d 上被强制分配的文字都替换为他们的前因，如果替换成的文字还是在决策级别 d 上被强制分配的，那就继续替换为他的前因，一直往前卷；当 w(i-1)中的所有文字都不是被强制赋值的，即表面到达固定点，则为学习到的新子句。

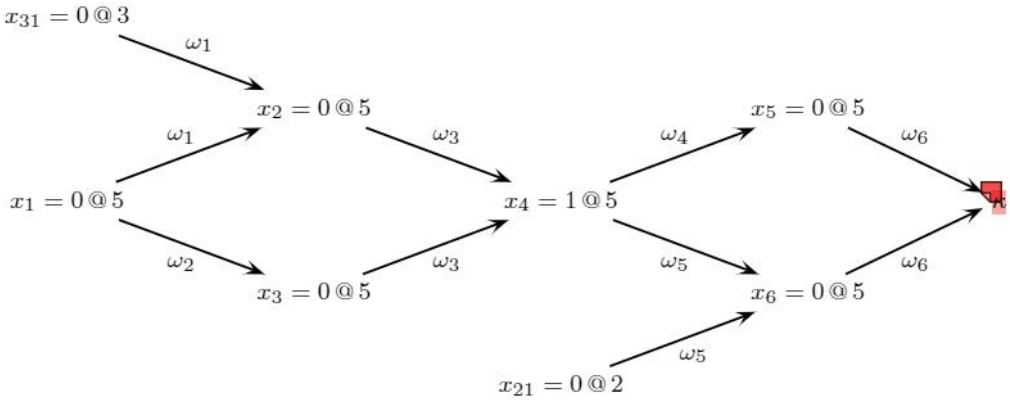


Figure 4.2. Implication graph for example 4.2.4

Table 4.1. Resolution steps during clause learning

$\omega_L^{5,0} = \{x_5, x_6\}$	Literals in $\alpha(\kappa)$
$\omega_L^{5,1} = \{\neg x_4, x_6\}$	Resolve with $\alpha(x_5) = \omega_4$
$\omega_L^{5,2} = \{\neg x_4, x_{21}\}$	Resolve with $\alpha(x_6) = \omega_5$
$\omega_L^{5,3} = \{x_2, x_3, x_{21}\}$	Resolve with $\alpha(x_4) = \omega_3$
$\omega_L^{5,4} = \{x_1, x_{31}, x_3, x_{21}\}$	Resolve with $\alpha(x_2) = \omega_1$
$\omega_L^{5,5} = \{x_1, x_{31}, x_{21}\}$	Resolve with $\alpha(x_3) = \omega_2$
$\omega_L^{5,6} = \{x_1, x_{31}, x_{21}\}$	No more resolution operations given (4.16)

利用 UIP 的结构

UIP 是蕴涵图中的支配者，表示在当前决策级别导致相同冲突的替代决策分配。识别 UIP 的主要动机是减少学习子句的大小。

在蕴涵图中，在决策层 d 有一个 UIP，当在决策层 d 分配的 $\omega_L^{d,i}$ 中的文字数为 1。令 σ(ω, d) 为子句 w 中在决策级别 d 上被分配的文字数：

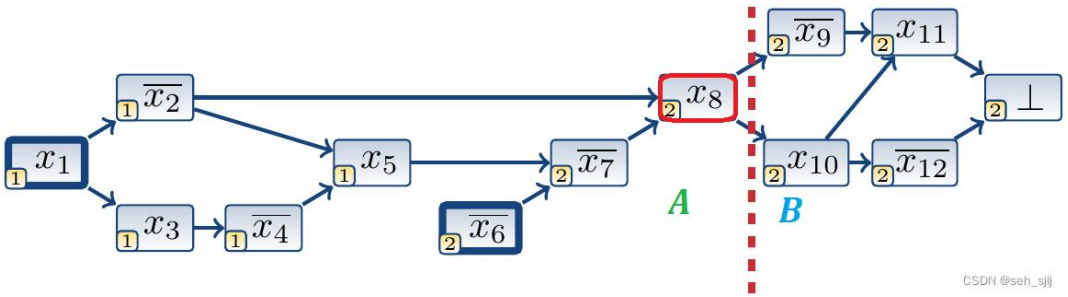
$$\sigma(\omega, d) = |\{l \in \omega \mid \delta(l) = d\}|$$

当 w(i-1)中的文字中只有一个是在决策级别 d 上被分配的，停止往前卷。

Table 4.2. Resolution steps during clause learning with UIPs

$\omega_L^{5,0} = \{x_5, x_6\}$	Literals in $\alpha(\kappa)$
$\omega_L^{5,1} = \{\neg x_4, x_6\}$	Resolve with $\alpha(x_5) = \omega_4$
$\omega_L^{5,2} = \{\neg x_4, x_{21}\}$	No more resolution operations given (4.18)

唯一蕴含点（unique implication point, UIP）：如果蕴含图G_π中的某个节点l是唯一蕴含点，那么设d是最近一次做决策的节点（即包含决策层最深的决策文字的节点），d到冲突节点的所有路径都必须经过l。例如上图中$d=\overline{x_6}$，冲突节点为⊥，l可以是$\overline{x_6}$，$\overline{x_7}$或x_8。



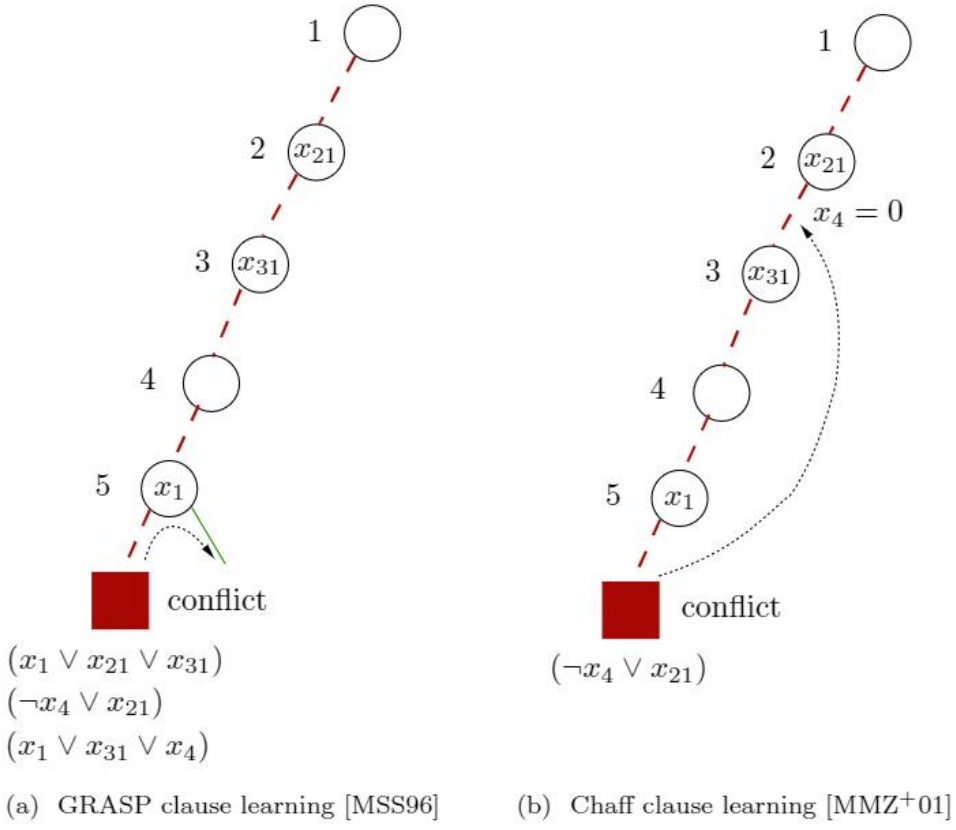
回溯方案

在 GRASP 学习和回溯方案中，对于第一次冲突，保留冲突的决策级别（即决策级别 5）。仅当结果单元传播产生另一个冲突时才会发生回溯。

GRASP 学习所有 UIP 的子句。 此外，GRASP 还学习了一个全局子句，然后将其用于在当前决策级别（在本例中为 5）强制执行第二个分支或用于回溯。GRASP 中使用的全局子句是可选的，用于确保在给定的决策级别上，两个分支都是针对同一变量进行的。

在 Chaff 学习和回溯方案中，回溯步骤总是在每次冲突后进行。由于 Chaff 在第一个 UIP 处停止，因此只学习了一个子句。

对于图中的示例，GRASP 将从决策级别 5 开始，而 Chaff 回溯到决策级别 2，并从决策级别 2 开始。



https://blog.csdn.net/qaqwqaqwq/article/details/126020807

https://blog.csdn.net/Pawa1uoke/article/details/121181101

ITE 操作是一个三元布尔操作符，对于具有相同变量序的三个布尔函数 f、g 和 h，ITE 操作可用来实现：if f then g else h。