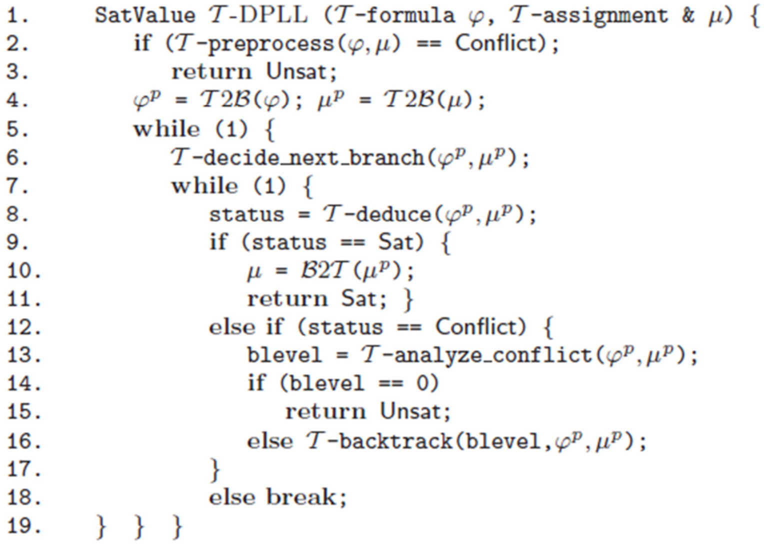
T-DPLL: Online Schema



```
SatValue $\mathcal{T}$-DPLL ($\mathcal{T}$-formula $\varphi$, $\mathcal{T}$-assignment & $\mu$) {
   if ($\mathcal{T}$-preprocess($\varphi, \mu$) == Conflict);
      return Unsat;
   $\varphi^p$ = $\mathcal{T}2\mathcal{B}(\varphi)$; $\mu^p$ = $\mathcal{T}2\mathcal{B}(\mu)$;
   while (1) {
      $\mathcal{T}$-decide_next_branch($\varphi^p, \mu^p$);
      while (1) {
         status = $\mathcal{T}$-deduce($\varphi^p, \mu^p$);
         if (status == Sat) {
            $\mu$ = $\mathcal{B}2\mathcal{T}(\mu^p)$;
            return Sat; }
         else if (status == Conflict) {
            blevel = $\mathcal{T}$-analyze_conflict($\varphi^p, \mu^p$);
            if (blevel == 0)
               return Unsat;
            else $\mathcal{T}$-backtrack(blevel, $\varphi^p, \mu^p$);
         }
         else break;
} } }
```

Input:

Original T-formula ϕ

Reference to a T-Assignment μ (a set of T-literals) which is initially empty

T-preprocess:

Simplifies ϕ into a simpler version (ϕ^P) using

- Boolean preprocessing from DPLL
- Theory-dependent rewriting steps on T-literals of ϕ

Updates μ if necessary

Returns UNSAT if a conflict is encountered

T-decide-next-branch

Selects literal to branch on as in DPLL

T-deduce

Iteratively deduces the implied Boolean literals and updates ϕ^P and μ^P until:

1. μ^P violates ϕ^P (and entails { [] }) – return Conflict
2. μ^P satisfies ϕ^P, and the result is run through the T-solver
   - satisfied by T-solver? return Sat
   - unsatisfied by T-solver? return Conflict
3. Nothing new can be deduced – return Unknown

T-analyze-conflict

Determine the Boolean conflict set, n^P; add it to ϕ^P, then backtrack up to blevel