AAA528: Computational Logic

Lecture 5 — DPLL(T)

Hakjoo Oh
2018 Fall

Deciding T-Satisfiability

- Equality

$$(x_1 = x_2 \vee x_1 = x_3) \wedge (x_1 = x_2 \vee x_1 = x_4) \wedge \\ x_1 \neq x_2 \wedge x_1 \neq x_3 \wedge x_1 \neq x_4$$

- Linear arithmetic

$$(x_1 + 2x_3 < 5) \vee \neg(x_3 \leq 1) \wedge (x_1 \geq 3)$$

- Arrays

$$(i = j \wedge a[j] = 1) \wedge \neg(a[i] = 1)$$

DPLL(T)

- Generalization of CDCL to decidable quantifier-free first-order theories
- Based on an interplay between a CDCL SAT solver and a decision procedure $\boldsymbol{DP_T}$ for the conjunctive fragment of $\boldsymbol{T}$
- Implemented in most Satisfiability Module Theory (SMT) solvers

Theory Solver (DP_T)

- A theory solver DP_T accepts conjunctive quantifier-free Σ-formulas, where conjunctive Σ-formulas are conjunctions of Σ-literals.
- This does not restrict the scope of the decision procedures. For arbitrary quantifier-free Σ-formula F, we can convert it to DNF:

$$F_1 \vee F_2 \vee \cdots \vee F_k.$$

F is T-satisfiable iff at least one F_i is T-satisfiable. E.g.,

$$\begin{array}{l}
(x_1 = x_2 \wedge x_1 = x_2 \wedge x_1 \neq x_2 \wedge x_1 \neq x_3 \wedge x_1 \neq x_4) \vee \\
(x_1 = x_2 \wedge x_1 = x_4 \wedge x_1 \neq x_2 \wedge x_1 \neq x_3 \wedge x_1 \neq x_4) \vee \\
(x_1 = x_3 \wedge x_1 = x_2 \wedge x_1 \neq x_2 \wedge x_1 \neq x_3 \wedge x_1 \neq x_4) \vee \\
(x_1 = x_3 \wedge x_1 = x_4 \wedge x_1 \neq x_2 \wedge x_1 \neq x_3 \wedge x_1 \neq x_4)
\end{array}$$

- This method misses any opportunity for learning, as each clause is solved independently, e.g., $x_1 = x_2 \wedge x_1 \neq x_2$.
- A better approach is to leverage the learning capabilities of SAT and combine it with DP_T.

DP_T for Equality Theory

- A literal is either equality or inequality.
- Given a conjunction of T-literals ϕ, build an undirected graph $G(N, E_=, E_{\neq})$.
  - V is the set of variables in ϕ.
  - $(x_1, x_2) \in E_=$ iff $(x_1 = x_2) \in \phi$
  - $(x_1, x_2) \in E_{\neq}$ iff $(x_1 \neq x_2) \in \phi$
- ϕ is unsatisfiable iff there exists an edge $(x_1, x_2) \in E_{\neq}$ such that x_2 is reachable from x_1 through a sequence of $E_=$ edges.
- Example:

$$x_1 \neq x_2 \wedge x_2 = x_3 \wedge x_1 = x_3$$

Overview of DPLL(T)

- Let $at(\phi)$ be the set of Σ-atoms in a given NNF formula ϕ.
- Assuming some order, let $at_i(\phi)$ be the i-th distinct atom in ϕ.
- Given an atom a, let $e(a)$ be the unique boolean variable associated with a (called boolean encoder of a). Given a Σ-formula t, let $e(t)$ be the boolean formula where each atom is replaced by $e(a)$.
- For example, if $x = y$ is a Σ-atom, then $e(x = y)$ denotes its encoder (boolean variable) and when $\phi := x = y \vee x = z$, $e(\phi) := e(x = y) \vee e(x = z)$.
- $e(\phi)$ is called the propositional skeleton of ϕ.

Overview of DPLL(T)

- Example: $\phi := x = y \wedge ((y = z \wedge \neg(x = z)) \vee x = z)$.
- The propositional skeleton:

$$e(\phi) := e(x = y) \wedge ((e(y = z) \wedge \neg e(x = z)) \vee e(x = z))$$

- Let B be a boolean formula initially set to $e(\phi)$.
- Next, we check the satisfiability of B using a SAT solver. A satisfying assignment

$$\alpha := \{e(x = y) \mapsto \textbf{true}, e(y = z) \mapsto \textbf{true}, e(x = z) \mapsto \textbf{false}\}$$

- This does not mean that ϕ is satisfiable. We need to check the conjunction:

$$\hat{Th}(\alpha) := x = y \wedge y = z \wedge \neg(x = z)$$

- $\hat{Th}(\alpha)$ is not satisfiable. We conjoin B with

$$e(\neg \hat{Th}(\alpha)) := (\neg e(x = y) \vee \neg e(y = z) \vee e(x = z))$$

  which is called a blocking clause or lemma.
- The SAT solver is invoked again and suggests another assignment:

$$\alpha' := \{e(x = y) \mapsto \textbf{true}, e(y = z) \mapsto \textbf{true}, e(x = z) \mapsto \textbf{true}\}$$

- The corresponding $\hat{Th}(\alpha') := x = y \wedge y = z \wedge x = z$ is satisfiable, so ϕ is.

Theory Propagation

- An improvement to the procedure.
- Invoke DP_T after some or all partial assignments, rather than waiting for a full assignment.
- When the partial assignment is not contradictory, propagate implications due to the theory T back to the SAT solver.
- Consider the partial assignment:

$$\alpha := \{e(x = y) \mapsto \textbf{true}, e(y = z) \mapsto \textbf{true}\}$$

  and the corresponding formula fed to DP_T:

$$\hat{Th}(\alpha) := x = y \land y = z.$$

- DP_T infers that $x = z$ and informs the SAT solver that $e(x = z) \mapsto$ **true** is implied by the partial assignment.
- In addition to BCP, we now has theory propagation (TP). TP may lead to further BCP, which means that (BCP,TP) may iterate several times.

DPLL(T)

Algorithm 11.2.3: DPLL(T)

Input: A formula φ
Output: "Satisfiable" if the formula is satisfiable and "Unsatisfiable" otherwise

```
function DPLL(T)
    AddClauses(cnf(e(φ)));
    if BCP() = "conflict" then return "Unsatisfiable";
    while (TRUE) do
        if ¬Decide() then return "Satisfiable";   ▷ Full assignment
        repeat
            while (BCP() = "conflict") do
                backtrack-level := Analyze-Conflict();
                if backtrack-level < 0 then return "Unsatisfiable";
                else BackTrack(backtrack-level);
            ⟨t, res⟩:=Deduction(T̂h(α));
            AddClauses(e(t));
        until t ≡ TRUE
```