2 Satisfiability Modulo Theories (SMT)

So far:

decision procedures for satisfiability for various fragments of first-order theories;

often only for ground conjunctions of literals.

Goals:

extend decision procedures efficiently to ground CNF formulas;

later: extend to non-ground formulas (we will often lose completeness, however).

2.1 The DPLL(T) Procedure

Goal:

Given a propositional formula in CNF (or alternatively, a finite set N of clauses), where the atoms represent ground formulas over some theory $\mathcal{T}$, check whether it is satisfiable in $\mathcal{T}$ (and optionally: output *one* solution, if it is satisfiable).

Assumption:

As in the propositional case, clauses contain neither duplicated literals nor complementary literals.

For propositional DPLL, we have considered partial valuations, i. e., partial mappings from propositional variables to truth values.

A partial valuation $\mathcal{A}$ corresponds to a set M of literals that does not contain complementary literals, and vice versa:

$\mathcal{A}(L)$ is true, if $L \in M$.

$\mathcal{A}(L)$ is false, if $\overline{L} \in M$.

$\mathcal{A}(L)$ is undefined, if neither $L \in M$ nor $\overline{L} \in M$.

We will now consider partial mappings from ground $\mathcal{T}$-atoms to truth values (which correspond to sets of $\mathcal{T}$-literals).

In order to check whether a (partial) valuation is permissible, we identify the valuation $\mathcal{A}$ or the set M with the conjunction of all literals in M:

The valuation $\mathcal{A}$ or the set M is called $\mathcal{T}$-satisfiable, if the literals in M have a $\mathcal{T}$-model.

Since the elements of M can be interpreted both as propositional variables and as ground $\mathcal{T}$-formulas, we have to distinguish between two notions of entailment:

We write $M \models F$ if F is entailed by M propositionally. We write $M \models_{\mathcal{T}} F$ if the ground $\mathcal{T}$-formulas represented by M entail F.

M is called a $\mathcal{T}$-model of F, if it is $\mathcal{T}$-satisfiable and $M \models F$.

We write $F \models_{\mathcal{T}} G$, if the formula F entails G w. r. t. $\mathcal{T}$, that is, if every $\mathcal{T}$-model of F is also a model of G.

Idea

Naive Approach:

Use DPLL to find a propositionally satisfying valuation.

If the valuation found is $\mathcal{T}$-satisfiable, stop; otherwise continue DPLL search.

Note: The DPLL procedure may *not* use "pure literal" checks.

Improvements:

Check already partial valuations for $\mathcal{T}$-satisfiability.

If $\mathcal{T}$-decision procedure yields explanations, use them for non-chronological backjumping.

If $\mathcal{T}$-decision procedure can provide $\mathcal{T}$-entailed literals, use them for propagation.

Since $\mathcal{T}$-satisfiability checks may be costly, learn clauses that incorporate useful $\mathcal{T}$-knowledge, in particular explanations for backjumping. (The procedure is also called CDCL, i. e., "conflict-driven clause learning".)

DPLL(T)

The "DPLL Modulo Theories" procedure is modelled by a transition relation $\Rightarrow_{\text{DPLL}(\mathcal{T})}$ on a set of states.

States:

- *fail*
- $M \parallel N$,

where M is a *list of annotated literals* and N is a set of clauses.

Annotated literal:

- L: deduced literal, due to unit propagation.
- L^{d}: decision literal (guessed literal).

DPLL(T) Rules from DPLL

Unit Propagate:

$$M \parallel N \cup \{C \vee L\} \;\Rightarrow_{\mathrm{DPLL}(\mathcal{T})}\; M\, L \parallel N \cup \{C \vee L\}$$

if C is false under M and L is undefined under M.

Decide:

$$M \parallel N \;\Rightarrow_{\mathrm{DPLL}(\mathcal{T})}\; M\, L^{\mathrm{d}} \parallel N$$

if L is undefined under M.

Fail:

$$M \parallel N \cup \{C\} \;\Rightarrow_{\mathrm{DPLL}(\mathcal{T})}\; \mathit{fail}$$

if C is false under M and M contains no decision literals.

Specific DPLL(T) Rules

$\mathcal{T}$-Backjump:

$$M\, L^{\mathrm{d}}\, M' \parallel N \;\Rightarrow_{\mathrm{DPLL}(\mathcal{T})}\; M\, L' \parallel N$$

if $M\, L^{\mathrm{d}}\, M' \models \neg C$ for some $C \in N$
and if there is some "backjump clause" $C' \vee L'$ such that
$N \models_{\mathcal{T}} C' \vee L'$ and $M \models \neg C'$,
L' is undefined under M, and
L' or $\overline{L'}$ occurs in N or in $M\, L^{\mathrm{d}}\, M'$.

$\mathcal{T}$-Learn:

$$M \parallel N \;\Rightarrow_{\mathrm{DPLL}(\mathcal{T})}\; M \parallel N \cup \{C\}$$

if $N \models_{\mathcal{T}} C$ and each atom of C occurs in N or M.

$\mathcal{T}$-Forget:

$$M \parallel N \cup \{C\} \;\Rightarrow_{\mathrm{DPLL}(\mathcal{T})}\; M \parallel N$$

if $N \models_{\mathcal{T}} C$.

$\mathcal{T}$-Propagate:

$$M \parallel N \;\Rightarrow_{\mathrm{DPLL}(\mathcal{T})}\; M\, L \parallel N$$

if $M \models_{\mathcal{T}} L$ where L is undefined in M, and L or $\overline{L}$ occurs in N.

DPLL(T) Properties

The system DPLL($\mathcal{T}$) consists of the rules Decide, Fail, Unit Propagate, $\mathcal{T}$-Propagate, $\mathcal{T}$-Backjump, $\mathcal{T}$-Learn and $\mathcal{T}$-Forget.

Lemma 2.1 *If we reach a state $M \parallel N$ starting from $\emptyset \parallel N$, then:*

(1) M does not contain complementary literals.

(2) Every deduced literal L in M follows from $\mathcal{T}$, N and decision literals occurring before L in M.

Proof. By induction on the length of the derivation. □

Lemma 2.2 *If no clause is learned infinitely often, then every derivation starting from $\emptyset \parallel N$ terminates.*

Proof. Similar to the propositional case.

Lemma 2.3 *If $\emptyset \parallel N \Rightarrow^*_{\mathrm{DPLL}(\mathcal{T})} M \parallel N'$ and there is some conflicting clause in $M \parallel N'$, that is, $M \models \neg C$ for some clause C in N, then either Fail or $\mathcal{T}$-Backjump applies to $M \parallel N'$.*

Proof. Similar to the propositional case. □

Lemma 2.4 *If $\emptyset \parallel N \Rightarrow^*_{\mathrm{DPLL}(\mathcal{T})} M \parallel N'$ and M is $\mathcal{T}$-unsatisfiable, then either there is a conflicting clause in $M \parallel N'$, or else $\mathcal{T}$-Learn applies to $M \parallel N'$, generating a conflicting clause.*

Proof. If M is $\mathcal{T}$-unsatisfiable, then there exists a subsequence $(L_1, \dots, L_n)$ of M such that $\emptyset \models_{\mathcal{T}} \overline{L_1} \vee \dots \vee \overline{L_n}$. Hence the conflicting clause $\overline{L_1} \vee \dots \vee \overline{L_n}$ is either in $M \parallel N'$, or else it can be learned by one $\mathcal{T}$-Learn step. □

Theorem 2.5 *Consider a derivation $\emptyset \parallel N \Rightarrow^*_{\mathrm{DPLL}(\mathcal{T})} S$, where no more rules of the DPLL(T) procedure are applicable to S except $\mathcal{T}$-Learn or $\mathcal{T}$-Forget, and if S has the form $M \parallel N'$ then M is $\mathcal{T}$-satisfiable. Then*

(1) S is fail iff N is $\mathcal{T}$-unsatisfiable.

(2) If S has the form $M \parallel N'$, then M is a $\mathcal{T}$-model of N.

The Solver Interface

The general DPLL($\mathcal{T}$) procedure has to be connected to a "Solver" for $\mathcal{T}$, a theory module that performs *at least* $\mathcal{T}$-satisfiability checks.

The solver is initialized with a list of all literals occurring in the input of the DPLL($\mathcal{T}$) procedure.

Internally, it keeps a stack I of theory literals that is initially empty. The solver performs the following operations on I:

SetTrue(L: $\mathcal{T}$-Literal):

Check whether $I \cup \{L\}$ is consistent.

If no: return an explanation for $\overline{L}$, that is, a subset J of I such that $J \models_{\mathcal{T}} \overline{L}$.

If yes: push L on I.

Optionally: Return a list of literals that are $\mathcal{T}$-consequences of $I \cup \{L\}$ (and have not yet been detected before).

Note: Depending on $\mathcal{T}$, detecting (all) $\mathcal{T}$-consequences may be very cheap or very expensive.

Backtrack(n: $\mathbb{N}$):

Pop n literals from I.

Explanation(L: $\mathcal{T}$-Literal):

Return an explanation for L, that is, a subset J of I such that $J \models_{\mathcal{T}} L$.

We assume that L has been returned previously as a result of some SetTrue(L') operation. No literal of J may occur in I after L'.