

SAT and SMT Solving

Sarah Winkler

SS 2018

Department of Computer Science
University of Innsbruck

Outline

- Summary of Last Week
- Correctness of $DPLL(T)$
- Congruence Closure
- Some More Practical SMT

Summary of Last Week

Definitions

for formulas F and G and list of literals M :

- ▶ **theory T** is set of first-order logic formulas without free variables
- ▶ F is **T -consistent** (or **T -satisfiable**) if $F \wedge T$ is satisfiable in first-order sense
- ▶ F is **T -inconsistent** (or **T -unsatisfiable**) if not T -consistent
- ▶ $M = l_1, \dots, l_k$ is **T -consistent** if $l_1 \wedge \dots \wedge l_k$ is
- ▶ M is **T -model** of F if $M \models F$ and M is T -consistent
- ▶ F **entails G in T** (denoted $F \models_T G$) if $F \wedge \neg G$ is T -inconsistent
- ▶ F and G are **T -equivalent** (denoted $F \equiv_T G$) if $F \models_T G$ and $G \models_T F$

Definition (Theory of Equality)

theory of equality (EQ) uses binary predicate \approx and consists of axioms

$$\forall x (x \approx x) \quad \forall x y (x \approx y \rightarrow y \approx x) \quad \forall x y z (x \approx y \wedge y \approx z \rightarrow x \approx z)$$

Definition (Theory of Equality With Uninterpreted Functions)

EUF over set of function symbols \mathcal{F} consists of equality axioms:

$$\forall x (x \approx x) \quad \forall x y (x \approx y \rightarrow y \approx x) \quad \forall x y z (x \approx y \wedge y \approx z \rightarrow x \approx z)$$

plus for all $f/n \in \mathcal{F}$ {the **functional consistency axiom**:

$$\forall x_1 y_1 \dots x_n y_n (x_1 \approx y_1 \wedge \dots \wedge x_n \approx y_n \rightarrow f(x_1, \dots, x_n) \approx f(y_1, \dots, y_n))$$

Definition (DPLL(T) Transition Rules)

DPLL(T) consists of DPLL rules unit propagate, decide, fail, and restart plus

- ▶ **T -backjump** $M I^d N \parallel F, C \implies M I' \parallel F, C$
if $M I^d N \models \neg C$ and \exists clause $C' \vee I'$ such that
 - ▶ $F, C \models_T C' \vee I'$
 - ▶ $M \models \neg C'$ and I' is undefined in M , and I' or I'^c occurs in F or in $M I^d N$
- ▶ **T -learn** $M \parallel F \implies M \parallel F, C$
if $F \models_T C$ and all atoms of C occur in M or F
- ▶ **T -forget** $M \parallel F, C \implies M \parallel F$
if $F \models_T C$
- ▶ **T -propagate** $M \parallel F \implies M I \parallel F$
if $M \models_T I$, literal I or I^c occurs in F , and I is undefined in M

Naive Lazy Approach in DPLL(T)

- ▶ whenever state $M \parallel F$ is final wrt unit propagate, decide, fail, T -backjump: check T -consistency of M with T -solver
- ▶ if M is T -consistent then satisfiability is proven
- ▶ otherwise $\exists l_1, \dots, l_k$ subset of M such that $\models_T \neg(l_1 \wedge \dots \wedge l_k)$
- ▶ use T -learn to add $\neg l_1 \vee \dots \vee \neg l_k$
- ▶ apply restart

Improvement 1: Incremental T -Solver

- ▶ T -solver checks T -consistency of model M whenever literal is added to M

Improvement 2: On-Line SAT solver

- ▶ after T -learn added clause, apply fail or T -backjump instead of restart

Improvement 3: Eager Theory Propagation

- ▶ apply T -propagate before decide

Remark

all three improvements can be combined

Correctness of DPLL(T)

Definition (Basic DPLL(T))

system \mathcal{B} consists of unit propagate, decide, fail, T -backjump, and T -propagate

Definition (Full DPLL(T))

system \mathcal{F} extends \mathcal{B} by T -learn, T -forget, and restart

Lemma

if $\| F \Longrightarrow_{\mathcal{F}}^* M \| G$ then

- ▶ all atoms in M and G are atoms in F
- ▶ M does not contain complementary literals, and every literal at most once
- ▶ G is T -equivalent to F ($F \equiv_T G$)
- ▶ if $M = M_0 l_1^d M_1 l_2^d M_2 \dots l_k^d M_k$ with l_1, \dots, l_k all the decision literals then $F, l_1, \dots, l_i \models_T M_i$ for all $0 \leq i \leq k$

Consider derivation with final state S_n :

$$\| F \implies_{\mathcal{F}} S_1 \implies_{\mathcal{F}} S_2 \implies_{\mathcal{F}} \dots \implies_{\mathcal{F}} S_n$$

Theorem

if $S_n = \text{FailState}$ then F is T -unsatisfiable

Proof.

- ▶ must have $\| F \implies_{\mathcal{F}}^* M \| F' \xRightarrow{\text{fail}}_{\mathcal{F}} \text{FailState}$
such that M contains no decision literals and $M \models \neg C$ for some C in F'
- ▶ by Lemma before, $F' \models_T M$, so $F' \models_T \neg C$
- ▶ also have $F' \models_T C$ because C is in F' , so $F \equiv_T F'$ is unsatisfiable ■

Theorem

if $S_n = M \| F'$ and M is T -consistent then F is T -satisfiable and $M \models_T F$

Proof.

- ▶ S_n is final, so all literals of F' are defined in M (otherwise decide applicable)
- ▶ \nexists clause C in F' such that $M \models_T \neg C$ (otherwise backjump or fail applicable)
- ▶ by T -consistency $M \models_T F'$
- ▶ have $F \equiv_T F'$ so M does also T -satisfy F ■

Theorem (Termination)

$$\Gamma: \quad || F \Longrightarrow_{\mathcal{F}}^* S_0 \Longrightarrow_{\mathcal{F}}^* S_1 \Longrightarrow_{\mathcal{F}}^* \dots$$

is *finite* if

- ▶ there is *no infinite sub-derivation* of only *T-learn* and *T-forget* steps, and
- ▶ for every sub-derivation

$$S_i \xrightarrow{\text{restart}}_{\mathcal{F}} S_{i+1} \Longrightarrow_{\mathcal{F}}^* S_j \xrightarrow{\text{restart}}_{\mathcal{F}} S_{j+1} \Longrightarrow_{\mathcal{F}}^* S_k$$

with no restart steps in $S_{i+1} \Longrightarrow_{\mathcal{F}}^* S_j$ and $S_{j+1} \Longrightarrow_{\mathcal{F}}^* S_k$:

- ▶ there are more *B*-steps in $S_j \Longrightarrow_{\mathcal{F}}^* S_k$ than in $S_i \Longrightarrow_{\mathcal{F}}^* S_j$, or
- ▶ a clause is learned in $S_j \Longrightarrow_{\mathcal{F}}^* S_k$ that is never forgotten in Γ

Proof.

similar as for DPLL:

- ▶ restart is applied with increasing periodicity, or
- ▶ otherwise clause is learned (and there are only finitely many clauses)



Congruence Closure

Aim

build T -solver for EUF using congruence closure

Definitions

- ▶ signature \mathcal{F} function symbols with fixed arity
- ▶ variables \mathcal{V} $\mathcal{F} \cap \mathcal{V} = \emptyset$
- ▶ terms $\mathcal{T}(\mathcal{F}, \mathcal{V})$ smallest set such that
 - ▶ $\mathcal{V} \subseteq \mathcal{T}(\mathcal{F}, \mathcal{V})$
 - ▶ if $f \in \mathcal{F}$ has arity n and $t_i \in \mathcal{T}(\mathcal{F}, \mathcal{V})$ then $f(t_1, \dots, t_n) \in \mathcal{T}(\mathcal{F}, \mathcal{V})$
- ▶ subterms

$$Sub(t) = \begin{cases} \{t\} & \text{if } t \in \mathcal{V} \\ \{t\} \cup \bigcup_i Sub(t_i) & \text{if } t = f(t_1, \dots, t_n) \end{cases}$$

Example

- ▶ for $\mathcal{F} = \{f/1, g/2, a/0\}$ and $x, y \in \mathcal{V}$ have terms $a, f(x), f(a), g(x, f(y)), \dots$
- ▶ for $t = g(g(x, x), f(f(a)))$ have $Sub(t) = \{t, g(x, x), x, f(f(a)), f(a), a\}$

Congruence Closure

Input: set of equations E and equation $s \approx t$, both without variables

Output: *valid* ($E \vDash_T s \approx t$) or *invalid* ($E \vDash_T s \not\approx t$)

- 1 build congruence classes
 - (a) put different subterms of terms in $E \cup \{s \approx t\}$ in separate sets
 - (b) merge sets $\{\dots, t_1, \dots\}$ and $\{\dots, t_2, \dots\}$ for all $t_1 \approx t_2$ in E
 - (c) merge sets $\{\dots, f(t_1, \dots, t_n), \dots\}$ and $\{\dots, f(u_1, \dots, u_n), \dots\}$ if t_i and u_i belong to same set for all $1 \leq i \leq n$, repeatedly
- 1 if s and t belong to same set then return *valid* else return *invalid*

Example (1)

- ▶ given set of equations E

$$f(f(f(a))) \approx g(f(g(f(b)))) \quad f(g(f(b))) \approx f(a) \quad g(g(b)) \approx g(f(a)) \quad g(a) \approx b$$

and equation $f(a) \approx g(a)$

- ▶ sets

- | | |
|---------------------------|------------------------------------|
| 1. $\{a\}$ | 5. $\{f(f(a))\}$ |
| 2. $\{f(a), f(g(f(b)))\}$ | 6. $\{f(f(f(a))), g(f(g(f(b))))\}$ |
| 3. $\{b, g(a)\}$ | 7. $\{f(b)\}$ |
| 4. $\{g(b)\}$ | 8. $\{g(f(b))\}$ |

- ▶ conclusion: $E \models s \not\approx t$

Example (2)

- ▶ given set of equations E

$$f(f(f(a))) \approx a \qquad f(f(f(f(f(a)))))) \approx a$$

and equation $f(a) \approx a$

- ▶ $\{ a, f(a), f(f(a)), f(f(f(a))), f(f(f(f(a)))) \}$
- ▶ conclusion: $E \models s \approx t$

Ok, But How About a Solver for EUF?

Definition (Skolemization)

given formula φ with free variables x_1, \dots, x_n ,

$\hat{\varphi} = \varphi[x_1 \mapsto c_1, \dots, x_n \mapsto c_n]$ where c_1, \dots, c_n are fresh constants

Deciding Satisfiability of EUF Conjunctions

for EUF conjunction $\varphi = (\bigwedge P) \wedge (\bigwedge \neg N)$, split into positive and negative literals with free variables x_1, \dots, x_n

$\varphi = (\bigwedge P) \wedge (\bigwedge \neg N)$	unsatisfiable	
$\iff \exists x_1 \dots x_n. (\bigwedge P) \wedge (\bigwedge \neg N)$	unsatisfiable	
$\iff (\bigwedge \hat{P}) \wedge (\bigwedge \neg \hat{N})$	unsatisfiable	skolemization
$\iff \neg \left((\bigwedge \hat{P}) \wedge (\bigwedge \neg \hat{N}) \right)$	valid	φ unsat iff $\neg\varphi$ valid
$\iff \bigwedge \hat{P} \rightarrow \bigvee \hat{N}$	valid	deMorgan
$\iff \exists s \approx t \text{ in } \hat{N} \text{ such that } \bigwedge \hat{P} \rightarrow s \approx t$	valid	semantics of \bigvee
$\iff \exists s \approx t \text{ in } \hat{N} \text{ such that } \bigwedge \hat{P} \models_{\mathcal{T}} s \approx t$		semantics of \exists

Obtained Satisfiability Check

$$(\bigwedge P) \wedge (\bigwedge \neg N) \text{ unsatisfiable} \iff \exists s \approx t \text{ in } \hat{N} \text{ such that } \bigwedge \hat{P} \models_{\mathcal{T}} s \approx t$$

Example

1 $g(a) \approx c \wedge f(g(a)) \not\approx f(c) \wedge c \not\approx d$

- ▶ split into $P = \{g(a) \approx c\}$ and $N = \{f(g(a)) \approx f(c), c \approx d\}$
- ▶ have $g(a) \approx c \models_{\mathcal{T}} f(g(a)) \approx f(c)$, so **unsatisfiable**

2 $g(a) \approx c \wedge f(g(a)) \approx f(c) \wedge g(a) \approx d \wedge c \not\approx d$

- ▶ split into $P = \{g(a) \approx c, f(g(a)) \approx f(c), g(a) \approx d\}$ and $N = \{c \approx d\}$
- ▶ have $g(a) \approx c, f(g(a)) \approx f(c), g(a) \approx d \models_{\mathcal{T}} c \approx d$, so **unsatisfiable**

3 $g(a) \approx c \wedge c \approx d \wedge f(x) \approx x \wedge d \not\approx g(x) \wedge f(x) \not\approx d$

- ▶ $P = \{g(a) \approx c, c \approx d, f(x) \approx x\}$ and $N = \{d \approx g(x), f(x) \approx d\}$
- ▶ **skolemize** $P = \{g(a) \approx c, c \approx d, f(e) \approx e\}$, $N = \{d \approx g(e), f(e) \approx d\}$
 - ▶ $g(a) \approx c, c \approx d, f(e) \approx e \not\models_{\mathcal{T}} d \approx g(e)$
 - ▶ $g(a) \approx c, c \approx d, f(e) \approx e \not\models_{\mathcal{T}} f(e) \approx d$

so **satisfiable**

Some More Practical SMT

Integer Arithmetic in python/z3

```
from z3 import *
a = Int('a') # create integer variables
b = Int('b')
c = Int('c')

phi = And(c > 0, b >= 0, a < -1)
psi = (a == If (b == c, b - 2, c - 4))
print(phi)
solver = Solver()
solver.add(phi, psi) # assert constraints
solver.add(a + b < 2 * c)

result = solver.check() # check for satisfiability
if result == z3.sat:
    model = solver.model() # get valuation
    print model[a], model[b], model[c] # -3 0 1
```