
# SAT and SMT Solving

**Sarah Winkler**

Computational Logic Group
Department of Computer Science
University of Innsbruck

lecture 6
SS 2019

## Outline

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

## 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 $l_1 \wedge \cdots \wedge l_k$ is
- $M$ is $T$-model of $F$ if $M \vDash F$ and $M$ is $T$-consistent
- $F$ entails $G$ in $T$ (denoted $F \vDash_T G$) if $F \wedge \neg G$ is $T$-inconsistent
- $F$ and $G$ are $T$-equivalent (denoted $F \equiv_T G$) if $F \vDash_T G$ and $G \vDash_T F$

**Definition (Theory of Equality)**

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

$$\forall x\ (x \approx x) \qquad \forall x\, y\ (x \approx y \ \rightarrow\ y \approx x) \qquad \forall x\, y\, z\ (x \approx y \wedge y \approx z \ \rightarrow\ x \approx z)$$

**Definition (Theory of Equality With Uninterpreted Functions)**

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

$$\forall x\ (x \approx x) \qquad \forall x\, y\ (x \approx y \ \rightarrow\ y \approx x) \qquad \forall x\, y\, z\ (x \approx y \wedge y \approx z \ \rightarrow\ x \approx z)$$

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

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

**Definition (DPLL($T$) Transition Rules)**

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

- $T$-backjump $\quad M\ l^d\ N \parallel F, C \implies M\ l' \parallel F, C$

  if $M\ l^d\ N \models \neg C$ and $\exists$ clause $C' \vee l'$ such that
  - $F, C \models_T C' \vee l'$
  - $M \models \neg C'$ and $l'$ is undefined in $M$, and $l'$ or $l'^c$ occurs in $F$ or in $M\ l^d\ N$

- $T$-learn $\quad M \parallel F \implies M \parallel F,\ C$

  if $F \models_T C$ and all atoms of $C$ occur in $M$ or $F$

- $T$-forget $\quad M \parallel F,\ C \implies M \parallel F$

  if $F \models_T C$

- $T$-propagate $\quad M \parallel F \implies M\ l \parallel F$

  if $M \models_T l$, literal $l$ or $l^c$ occurs in $F$, and $l$ 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 $\exists l_1, \ldots, l_k$ subset of $M$ such that $\vDash_T \neg(l_1 \wedge \cdots \wedge l_k)$
- use $T$-learn to add $\neg l_1 \vee \cdots \vee \neg l_k$
- 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

# Outline

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

**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* $\parallel F \Longrightarrow^*_{\mathsf{F}} M \parallel G$ *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* $F,\ l_1, \dots, l_i \vDash_T M_i$ *for all* $0 \leqslant i \leqslant k$

Consider derivation with final state $S_n$:

$$\| F \quad \Longrightarrow_{\mathcal{F}} \quad S_1 \quad \Longrightarrow_{\mathcal{F}} \quad S_2 \quad \Longrightarrow_{\mathcal{F}} \quad \dots \quad \Longrightarrow_{\mathcal{F}} \quad S_n$$

**Theorem**

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

**Proof.**

- must have $\| F \Longrightarrow_{\mathcal{F}}^{*} M \| F' \stackrel{\text{fail}}{\Longrightarrow}_{\mathcal{F}} \text{FailState}$, so $M \models \neg C$ for some $C$ in $F'$
- $M$ cannot contain decision literals (otherwise $T$-backjump applicable)
- 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 $T$-inconsistent ∎

**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 $M \models \neg C$ (otherwise backjump or fail applicable)
- so $M \models F'$ and by $T$-consistency $M \models_T F'$
- have $F \equiv_T F'$ so $M$ also $T$-satisfies $F$ ∎

**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*

$$S_i \overset{restart}{\Longrightarrow}_{\mathcal{F}} S_{i+1} \Longrightarrow^*_{\mathcal{F}} S_j \overset{restart}{\Longrightarrow}_{\mathcal{F}} S_{j+1} \Longrightarrow^*_{\mathcal{F}} S_k \overset{restart}{\Longrightarrow}_{\mathcal{F}} S_{k+1}$$

  *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 $\mathcal{B}$-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 $\Gamma$*

**Proof.**

similar as for DPLL:

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

## Outline

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

## Aim

build $T$-solver for EUF using congruence closure

## Definitions (Terms)

number of arguments

- signature $\qquad \mathcal{F} \qquad$ function symbols with fixed arity
- variables $\qquad \mathcal{V} \qquad \mathcal{F} \cap \mathcal{V} = \varnothing$
- terms $\qquad \mathcal{T}(\mathcal{F}, \mathcal{V}) \qquad$ 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

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

## Example

- for $\mathcal{F} = \{\mathsf{f}/1,\ \mathsf{g}/2,\ \mathsf{a}/0\}$ and $x, y \in \mathcal{V}$ have terms $\mathsf{a}$, $\mathsf{f}(x)$, $\mathsf{f}(\mathsf{a})$, $\mathsf{g}(x, \mathsf{f}(y))$, ...
- for $t = \mathsf{g}(\mathsf{g}(x,x), \mathsf{f}(\mathsf{f}(\mathsf{a})))$ have $\mathcal{S}ub(t) = \{t,\ \mathsf{g}(x,x),\ x,\ \mathsf{f}(\mathsf{f}(\mathsf{a})),\ \mathsf{f}(\mathsf{a}),\ \mathsf{a}\}$

## 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* ($E \nvDash_T s \approx t$)

1. build congruence classes
   (a) put different subterms of $E \cup \{s \approx t\}$ 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 $1 \leqslant i \leqslant n$
   (d) repeat (c) until no change
1. if $s$ and $t$ belong to same set then return *valid* else return *invalid*

## Example (1)

- given set of equations $E$

  $$\mathsf{f}(\mathsf{f}(\mathsf{f}(\mathsf{a}))) \approx \mathsf{g}(\mathsf{f}(\mathsf{g}(\mathsf{f}(\mathsf{b})))) \quad \mathsf{f}(\mathsf{g}(\mathsf{f}(\mathsf{b}))) \approx \mathsf{f}(\mathsf{a}) \quad \mathsf{g}(\mathsf{g}(\mathsf{b})) \approx \mathsf{g}(\mathsf{f}(\mathsf{a})) \quad \mathsf{g}(\mathsf{a}) \approx \mathsf{b}$$

  and test equation $\mathsf{f}(\mathsf{a}) \approx \mathsf{g}(\mathsf{a})$

- sets

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

- conclusion: $E \vDash \mathsf{f}(\mathsf{a}) \not\approx \mathsf{g}(\mathsf{a})$

## Example (2)

- given set of equations $E$

$$\mathsf{f}(\mathsf{f}(\mathsf{f}(\mathsf{a}))) \approx \mathsf{a} \qquad \mathsf{f}(\mathsf{f}(\mathsf{f}(\mathsf{f}(\mathsf{f}(\mathsf{a}))))) \approx \mathsf{a}$$

  and test equaton $\mathsf{f}(\mathsf{a}) \approx \mathsf{a}$

- $\{\, \mathsf{a}, \mathsf{f}(\mathsf{a}), \mathsf{f}(\mathsf{f}(\mathsf{a})), \mathsf{f}(\mathsf{f}(\mathsf{f}(\mathsf{a}))), \mathsf{f}(\mathsf{f}(\mathsf{f}(\mathsf{f}(\mathsf{a})))), \mathsf{f}(\mathsf{f}(\mathsf{f}(\mathsf{f}(\mathsf{f}(\mathsf{a}))))) \,\}$

- conclusion: $E \vDash \mathsf{f}(\mathsf{a}) \approx \mathsf{a}$

# Ok, But How About a Solver for EUF?

## Definition (Skolemization)

given formula $\varphi$ with free variables $x_1, \dots, x_n$,
$\widehat{\varphi} = \varphi[x_1 \mapsto c_1,\ \dots, x_n \mapsto c_n]$ where $c_1, \dots, c_n$ are fresh constants

## Deciding Satisfiability of EUF Conjunctions

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

$$
\begin{array}{lll}
\varphi = (\bigwedge P) \wedge (\bigwedge \neg N) & \text{unsatisfiable} & \\
\quad\iff \exists x_1 \dots x_n. (\bigwedge P) \wedge (\bigwedge \neg N) & \text{unsatisfiable} & \\
\quad\iff (\bigwedge \widehat{P}) \wedge (\bigwedge \neg \widehat{N}) & \text{unsatisfiable} & \text{skolemization} \\
\quad\iff \neg \Big( (\bigwedge \widehat{P}) \wedge (\bigwedge \neg \widehat{N}) \Big) & \text{valid} & \varphi \text{ unsat iff } \neg\varphi \text{ valid} \\
\quad\iff \bigwedge \widehat{P} \to \bigvee \widehat{N} & \text{valid} & \text{de Morgan} \\
\quad\iff \exists\, s \approx t \text{ in } \widehat{N} \text{ such that } \bigwedge \widehat{P} \to s \approx t \text{ valid} & & \text{semantics of } \vee \\
\quad\iff \exists\, s \approx t \text{ in } \widehat{N} \text{ such that } \bigwedge \widehat{P} \vDash_T s \approx t & & \text{semantics of } \vDash
\end{array}
$$

## Obtained Satisfiability Check

$$(\bigwedge P) \land (\bigwedge \neg N) \text{ unsatisfiable} \quad \Longleftrightarrow \quad \exists\, s \approx t \text{ in } \widehat{N} \text{ such that } \bigwedge \widehat{P} \models_T s \approx t$$

## Example

1. $\mathsf{g}(\mathsf{a}) \approx \mathsf{c} \land \mathsf{f}(\mathsf{g}(\mathsf{a})) \not\approx \mathsf{f}(\mathsf{c}) \land \mathsf{c} \not\approx \mathsf{d}$
   - split into $P = \{\mathsf{g}(\mathsf{a}) \approx \mathsf{c}\}$ and $N = \{\mathsf{f}(\mathsf{g}(\mathsf{a})) \approx \mathsf{f}(\mathsf{c}),\ \mathsf{c} \approx \mathsf{d}\}$
   - have $\mathsf{g}(\mathsf{a}) \approx \mathsf{c} \models_T \mathsf{f}(\mathsf{g}(\mathsf{a})) \approx \mathsf{f}(\mathsf{c})$, so unsatisfiable

2. $\mathsf{g}(\mathsf{a}) \approx \mathsf{c} \land \mathsf{f}(\mathsf{g}(\mathsf{a})) \approx \mathsf{f}(\mathsf{c}) \land \mathsf{g}(\mathsf{a}) \approx \mathsf{d} \land \mathsf{c} \not\approx \mathsf{d}$
   - split into $P = \{\mathsf{g}(\mathsf{a}) \approx \mathsf{c},\ \mathsf{f}(\mathsf{g}(\mathsf{a})) \approx \mathsf{f}(\mathsf{c}),\ \mathsf{g}(\mathsf{a}) \approx \mathsf{d}\}$ and $N = \{\mathsf{c} \approx \mathsf{d}\}$
   - have $\mathsf{g}(\mathsf{a}) \approx \mathsf{c}, \mathsf{f}(\mathsf{g}(\mathsf{a})) \approx \mathsf{f}(\mathsf{c}), \mathsf{g}(\mathsf{a}) \approx \mathsf{d} \models_T \mathsf{c} \approx \mathsf{d}$, so unsatisfiable

3. $\mathsf{g}(\mathsf{a}) \approx \mathsf{c} \land \mathsf{c} \approx \mathsf{d} \land \mathsf{f}(x) \approx x \land \mathsf{d} \not\approx \mathsf{g}(x) \land \mathsf{f}(x) \not\approx \mathsf{d}$
   - $P = \{\mathsf{g}(\mathsf{a}) \approx \mathsf{c},\ \mathsf{c} \approx \mathsf{d},\ \mathsf{f}(x) \approx x\}$ and $N = \{\mathsf{d} \approx \mathsf{g}(x),\ \mathsf{f}(x) \approx \mathsf{d}\}$
   - skolemize $P = \{\mathsf{g}(\mathsf{a}) \approx \mathsf{c},\ \mathsf{c} \approx \mathsf{d},\ \mathsf{f}(\mathsf{e}) \approx \mathsf{e}\}$, $N = \{\mathsf{d} \approx \mathsf{g}(\mathsf{e}),\ \mathsf{f}(\mathsf{e}) \approx \mathsf{d}\}$
     - $\mathsf{g}(\mathsf{a}) \approx \mathsf{c},\ \mathsf{c} \approx \mathsf{d},\ \mathsf{f}(\mathsf{e}) \approx \mathsf{e} \not\models_T \mathsf{d} \approx \mathsf{g}(\mathsf{e})$
     - $\mathsf{g}(\mathsf{a}) \approx \mathsf{c},\ \mathsf{c} \approx \mathsf{d},\ \mathsf{f}(\mathsf{e}) \approx \mathsf{e} \not\models_T \mathsf{f}(\mathsf{e}) \approx \mathsf{d}$

   so satisfiable

# 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) # some comparisons
psi = (a == If (b == c, b - 2, c - 4)) # if-then-else expression
print(phi)
solver = Solver()
solver.add(phi, psi) # assert constraints
solver.add(a + b < 2 * c) # arithmetic

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
```