Satisfiability Modulo Theories and Z3

Nikolaj Bjørner
Microsoft Research
ReRISE Winter School, Linz, Austria
February 4, 2014
Plan

Mon  An invitation to SMT with Z3

Tue  Equalities and Theory Combination

Wed  Theories: Arithmetic, Arrays, Data types

Thu  Quantifiers and Theories

Fri  Programming Z3: Interfacing and Solving
Lecture Overview

• Deciding Equality

• Uninterpreted Functions

• Nelson Oppen Combination

• Model-based Theory Combination
Deciding Equality

\[ a = b, \ b = c, \ d = e, \ b = s, \ d = t, \ a \neq e, \ a \neq s \]
Deciding Equality

\[ a = b, \ b = c, \ d = e, \ b = s, \ d = t, \ a \neq e, \ a \neq s \]
Deciding Equality

\[ a = b, \ b = c, \ d = e, \ b = s, \ d = t, \ a \neq e, \ a \neq s \]
Deciding Equality

\[ a = b, \ b = c, \ d = e, \ b = s, \ d = t, \ a \neq e, \ a \neq s \]
Deciding Equality

\[ a = b, \ b = c, \ d = e, \ b = s, \ d = t, \ a \neq e, \ a \neq s \]
Deciding Equality

\[a = b, \quad b = c, \quad d = e, \quad b = s, \quad d = t, \quad a \neq e, \quad a \neq s\]
Deciding Equality

\[ a = b, \ b = c, \ d = e, \ b = s, \ d = t, \ a \neq e, \ a \neq s \]
Deciding Equality

\[ a = b, \ b = c, \ d = e, \ b = s, \ d = t, \ a \neq e, \ a \neq s \]
Deciding Equality

\[ a = b, \ b = c, \ d = e, \ b = s, \ d = t, \ a \neq e, \ a \neq s \]
Deciding Equality

\[ a = b, \ b = c, \ d = e, \ b = s, \ d = t, \ a \neq e, \ a \neq s \]
Deciding Equality

\[ a = b, \quad b = c, \quad d = e, \quad b = s, \quad d = t, \quad a \neq e, \quad a \neq s \]
Deciding Equality

\[
a = b, \quad b = c, \quad d = e, \quad b = s, \quad d = t, \quad a \neq e, \quad a \neq s
\]
Deciding Equality

\[ a = b, b = c, d = e, b = s, d = t, a \neq e, a \neq s \]

Unsatisfiable
Deciding Equality

\[ a = b, \ b = c, \ d = e, \ b = s, \ d = t, \ a \neq e \]

Model construction

- \( \{a, b, c, s\} \)
- \( \{d, e, t\} \)
Deciding Equality

\[ a = b, \ b = c, \ d = e, \ b = s, \ d = t, \ a \neq e \]

Model construction

\[ |M| = \{\diamond_1, \diamond_2\} \] (universe, aka domain)
Deciding Equality

\[ a = b, \ b = c, \ d = e, \ b = s, \ d = t, \ a \neq e \]

Model construction

\[ |M| = \{\diamondsuit_1, \diamondsuit_2\} \] (universe, aka domain)

\[ M(a) = \diamondsuit_1 \] (assignment)
Deciding Equality

a = b, b = c, d = e, b = s, d = t, a \neq e

Model construction

\[ |M| = \{ \diamondsuit_1, \diamondsuit_2 \} \] (universe, aka domain)
\[ M(a) = \diamondsuit_1 \] (assignment)

Alternative notation:
\[ a^M = \diamondsuit_1 \]
Deciding Equality

\[ a = b, \ b = c, \ d = e, \ b = s, \ d = t, \ a \neq e \]

Model construction

\[ |M| = \{ \Diamond_1, \Diamond_2 \} \quad (\text{universe, aka domain}) \]

\[ M(a) = M(b) = M(c) = M(s) = \Diamond_1 \]

\[ M(d) = M(e) = M(t) = \Diamond_2 \]
Deciding Equality: Termination, Soundness, Completeness

• Termination: easy

• Soundness
  – Invariant: all constants in a “ball” are known to be equal.
  – The “ball” merge operation is justified by:
    • Transitivity and Symmetry rules.

• Completeness
  – We can build a model if an inconsistency was not detected.
  – Proof template (by contradiction):
    • Build a candidate model.
    • Assume a literal was not satisfied.
    • Find contradiction.
vector<int> F;

int new_node() { F.push_back(-1); return F.size()-1; }

int find(int node) {
    if (F[node] != -1) { F[node] = find(node); return F[node]; }
    return node;
}

void merge(int n1, int n2) {
    n1 = find(n1); n2 = find(n2);
    if (F[n1] > F[n2]) swap(n1, n2);
    if (n1 == n2) return;
    F[n1] += F[n2];
    F[n2] = n1;
}

- Size of equivalence class

Lazy path compression
Variant: Eager Path compression + equivalence class as doubly linked list

nlog*(n) amortized time for n operations

Root for largest class takes over
Deciding Equality: Termination, Soundness, Completeness

• Completeness
  – We can build a model if an inconsistency was not detected.
  – Instantiating the template for our procedure:
    • Assume some literal $c = d$ is not satisfied by our model.
    • That is, $M(c) \neq M(d)$.
    • This is impossible, $c$ and $d$ must be in the same “ball”.

$\downarrow_i \quad M(c) = M(d) = \downarrow_i \\
\downarrow_i \\
c, d, ...$
Completeness

We can build a model if an inconsistency was not detected.

Instantiating the template for our procedure:

Assume some literal \( c \neq d \) is not satisfied by our model.
That is, \( M(c) = M(d) \).

Key property: we only check the disequalities after we processed all equalities.

This is impossible, \( c \) and \( d \) must be in the different “balls”
Deciding Equality + (uninterpreted) Functions

\[ a = b, \ b = c, \ d = e, \ b = s, \ d = t, \ f(a, g(d)) \neq f(b, g(e)) \]

Congruence Rule:
\[ x_1 = y_1, \ldots, \ x_n = y_n \implies f(x_1, \ldots, x_n) = f(y_1, \ldots, y_n) \]
a = b, b = c, d = e, b = s, d = t, \( f(a, g(d)) \neq f(b, g(e)) \)

First Step: “Naming” subterms

Congruence Rule:
\[ x_1 = y_1, \ldots, x_n = y_n \implies f(x_1, \ldots, x_n) = f(y_1, \ldots, y_n) \]
Deciding Equality + (uninterpreted) Functions

\[ a = b, \ b = c, \ d = e, \ b = s, \ d = t, \ f(a, v_1) \neq f(b, g(e)) \]
\[ v_1 \equiv g(d) \]

First Step: “Naming” subterms

Congruence Rule:
\[ x_1 = y_1, ..., x_n = y_n \text{ implies } f(x_1, ..., x_n) = f(y_1, ..., y_n) \]
Deciding Equality + (uninterpreted) Functions

\[ a = b, \ b = c, \ d = e, \ b = s, \ d = t, \ f(a, \ v_1) \neq f(b, \ g(e)) \]
\[ v_1 \equiv g(d) \]

First Step: “Naming” subterms

Congruence Rule:
\[ x_1 = y_1, \ldots, x_n = y_n \text{ implies } f(x_1, \ldots, x_n) = f(y_1, \ldots, y_n) \]
Deciding Equality + (uninterpreted) Functions

\[ a = b, \; b = c, \; d = e, \; b = s, \; d = t, \; f(a, v_1) \neq f(b, v_2) \]
\[ v_1 \equiv g(d), \; v_2 \equiv g(e) \]

First Step: “Naming” subterms

Congruence Rule:
\[ x_1 = y_1, \; ..., \; x_n = y_n \implies f(x_1, ..., x_n) = f(y_1, ..., y_n) \]
Deciding Equality + (uninterpreted) Functions

\[ a = b, \; b = c, \; d = e, \; b = s, \; d = t, \; f(a, \; v_1) \neq f(b, \; v_2) \]

\[ v_1 \equiv g(d), \; v_2 \equiv g(e) \]

First Step: “Naming” subterms

Congruence Rule:
\[ x_1 = y_1, \; \ldots, \; x_n = y_n \text{ implies } f(x_1, \; \ldots, \; x_n) = f(y_1, \; \ldots, \; y_n) \]
Deciding Equality + (uninterpreted) Functions

\[ a = b, \ b = c, \ d = e, \ b = s, \ d = t, \ v_3 \neq f(b, \ v_2) \]

\[ v_1 \equiv g(d), \ v_2 \equiv g(e), \ v_3 \equiv f(a, \ v_1) \]

First Step: “Naming” subterms

Congruence Rule:

\[ x_1 = y_1, \ldots, \ x_n = y_n \] implies \[ f(x_1, \ldots, x_n) = f(y_1, \ldots, y_n) \]
Deciding Equality + (uninterpreted) Functions

\[ a = b, \ b = c, \ d = e, \ b = s, \ d = t, \ v_3 \neq f(b, v_2) \]

\[ v_1 \equiv g(d), \ v_2 \equiv g(e), \ v_3 \equiv f(a, v_1) \]

First Step: “Naming” subterms

Congruence Rule:

\[ x_1 = y_1, \ ..., \ x_n = y_n \text{ implies } f(x_1, ..., x_n) = f(y_1, ..., y_n) \]
Deciding Equality + (uninterpreted) Functions

\[ a = b, \quad b = c, \quad d = e, \quad b = s, \quad d = t, \quad v_3 \neq v_4 \]

\[ v_1 \equiv g(d), \quad v_2 \equiv g(e), \quad v_3 \equiv f(a, v_1), \quad v_4 \equiv f(b, v_2) \]

First Step: “Naming” subterms

Congruence Rule:

\[ x_1 = y_1, \ldots, x_n = y_n \text{ implies } f(x_1, \ldots, x_n) = f(y_1, \ldots, y_n) \]
Deciding Equality + (uninterpreted) Functions

\[ a = b, \ b = c, \ d = e, \ b = s, \ d = t, \ v_3 \neq v_4 \]

\[ v_1 \equiv g(d), \ v_2 \equiv g(e), \ v_3 \equiv f(a, v_1), \ v_4 \equiv f(b, v_2) \]

**Congruence Rule:**

\[ x_1 = y_1, \ldots, x_n = y_n \implies f(x_1, \ldots, x_n) = f(y_1, \ldots, y_n) \]
Deciding Equality + (uninterpreted) Functions

\[ a = b, \ b = c, \ d = e, \ b = s, \ d = t, \ v_3 \neq v_4 \]
\[ v_1 \equiv g(d), \ v_2 \equiv g(e), \ v_3 \equiv f(a, v_1), \ v_4 \equiv f(b, v_2) \]

**Congruence Rule:**

\[ x_1 = y_1, \ldots, x_n = y_n \text{ implies } f(x_1, \ldots, x_n) = f(y_1, \ldots, y_n) \]
\[ d = e \text{ implies } g(d) = g(e) \]
Deciding Equality + (uninterpreted) Functions

\[ a = b, \ b = c, \ d = e, \ b = s, \ d = t, \ \nu_3 \neq \nu_4 \]

\[ \nu_1 \equiv g(d), \ \nu_2 \equiv g(e), \ \nu_3 \equiv f(a, \ \nu_1), \ \nu_4 \equiv f(b, \ \nu_2) \]

**Congruence Rule:**

\[ x_1 = y_1, \ldots, x_n = y_n \text{ implies } f(x_1, \ldots, x_n) = f(y_1, \ldots, y_n) \]

\[ d = e \text{ implies } \nu_1 = \nu_2 \]
Deciding Equality for (uninterpreted) Functions

\[ a = b, \ b = c, \ d = e, \ b = s, \ d = t, \ v_3 \neq v_4 \]

\[ v_1 \equiv g(d), \ v_2 \equiv g(e), \ v_3 \equiv f(a, v_1), \ v_4 \equiv f(b, v_2) \]

Congruence Rule:

\[ x_1 = y_1, ..., x_n = y_n \text{ implies } f(x_1, ..., x_n) = f(y_1, ..., y_n) \]

\[ d = e \text{ implies } v_1 = v_2 \]
Deciding Equality + (uninterpreted) Functions

\[ a = b, \ b = c, \ d = e, \ b = s, \ d = t, \ v_3 \neq v_4 \]

\[ v_1 \equiv g(d), \ v_2 \equiv g(e), \ v_3 \equiv f(a, v_1), \ v_4 \equiv f(b, v_2) \]

**Congruence Rule:**

\[ x_1 = y_1, \ldots, x_n = y_n \text{ implies } f(x_1, \ldots, x_n) = f(y_1, \ldots, y_n) \]

\[ a = b, \ v_1 = v_2 \text{ implies } f(a, v_1) = f(b, v_2) \]
Deciding Equality + (uninterpreted) Functions

\[ a = b, \ b = c, \ d = e, \ b = s, \ d = t, \ v_3 \neq v_4 \]

\[ v_1 \equiv g(d), \ v_2 \equiv g(e), \ v_3 \equiv f(a, v_1), \ v_4 \equiv f(b, v_2) \]

Congruence Rule:

\[ x_1 = y_1, \ldots, x_n = y_n \text{ implies } f(x_1, \ldots, x_n) = f(y_1, \ldots, y_n) \]

\[ a = b, \ v_1 = v_2 \text{ implies } v_3 = v_4 \]
Deciding Equality + (uninterpreted) Functions

\[ a = b, \ b = c, \ d = e, \ b = s, \ d = t, \ v_3 \neq v_4 \]

\[ v_1 \equiv g(d), \ v_2 \equiv g(e), \ v_3 \equiv f(a, v_1), \ v_4 \equiv f(b, v_2) \]

Congruence Rule:

\[ x_1 = y_1, \ ldots, x_n = y_n \text{ implies } f(x_1, \ldots, x_n) = f(y_1, \ldots, y_n) \]

\[ a = b, \ v_1 = v_2 \text{ implies } v_3 = v_4 \]
Deciding Equality + (uninterpreted) Functions

\[ a = b, b = c, d = e, b = s, d = t, v_3 \neq v_4 \]

\[ v_1 \equiv g(d), v_2 \equiv g(e), v_3 \equiv f(a, v_1), v_4 \equiv f(b, v_2) \]

**Congruence Rule:**

\[ x_1 = y_1, \ldots, x_n = y_n \implies f(x_1, \ldots, x_n) = f(y_1, \ldots, y_n) \]
Deciding Equality + (uninterpreted) Functions

\[ a = b, \ b = c, \ d = e, \ b = s, \ d = t, \ a \neq v_4, \ v_2 \neq v_3 \]

\[ v_1 \equiv g(d), \ v_2 \equiv g(e), \ v_3 \equiv f(a, v_1), \ v_4 \equiv f(b, v_2) \]

Changing the problem

Congruence Rule:

\[ x_1 = y_1, \ ..., \ x_n = y_n \text{ implies } f(x_1, ..., x_n) = f(y_1, ..., y_n) \]
Deciding Equality + (uninterpreted) Functions

\[ a = b, \ b = c, \ d = e, \ b = s, \ d = t, \ a \neq v_4, \ v_2 \neq v_3 \]

\[ v_1 \equiv g(d), \ v_2 \equiv g(e), \ v_3 \equiv f(a, v_1), \ v_4 \equiv f(b, v_2) \]

Congruence Rule:

\[ x_1 = y_1, \ldots, \ x_n = y_n \implies f(x_1, \ldots, x_n) = f(y_1, \ldots, y_n) \]
Deciding Equality +
(uninterpreted) Functions

\[ a = b, \ b = c, \ d = e, \ b = s, \ d = t, \ a \neq v_4, \ V_2 \neq V_3 \]

\[ v_1 \equiv g(d), \ v_2 \equiv g(e), \ v_3 \equiv f(a, v_1), \ v_4 \equiv f(b, v_2) \]

Congruence Rule:

\[ x_1 = y_1, \ldots, x_n = y_n \text{ implies } f(x_1, \ldots, x_n) = f(y_1, \ldots, y_n) \]
Deciding Equality + (uninterpreted) Functions

$a = b, \ b = c, \ d = e, \ b = s, \ d = t, \ a \neq v_4, \ v_2 \neq v_3$

$v_1 \equiv g(d), \ v_2 \equiv g(e), \ v_3 \equiv f(a, v_1), \ v_4 \equiv f(b, v_2)$

Model construction:

$|M| = \{\smiley_1, \smiley_2, \smiley_3, \smiley_4\}$

$M(a) = M(b) = M(c) = M(s) = \smiley_1$

$M(d) = M(e) = M(t) = \smiley_2$

$M(v_1) = M(v_2) = \smiley_3$

$M(v_3) = M(v_4) = \smiley_4$
Deciding Equality + (uninterpreted) Functions

\[ a = b, b = c, d = e, b = s, d = t, a \neq v_4, v_2 \neq v_3 \]

\[ v_1 \equiv g(d), v_2 \equiv g(e), v_3 \equiv f(a, v_1), v_4 \equiv f(b, v_2) \]

Model construction:
\[
|M| = \{\spadesuit_1, \spadesuit_2, \spadesuit_3, \spadesuit_4\}
\]

\[
M(a) = M(b) = M(c) = M(s) = \spadesuit_1
\]
\[
M(d) = M(e) = M(t) = \spadesuit_2
\]
\[
M(v_1) = M(v_2) = \spadesuit_3
\]
\[
M(v_3) = M(v_4) = \spadesuit_4
\]

Missing: Interpretation for \( f \) and \( g \).
Deciding Equality + (uninterpreted) Functions

Building the interpretation for function symbols

- $M(g)$ is a mapping from $|M|$ to $|M|$.
- Defined as:
  
  $M(g)(\diamond_i) = \diamond_j$ if there is $v \equiv g(a)$ s.t.
  
  $M(a) = \diamond_i$
  $M(v) = \diamond_j$

  $= \diamond_k$, otherwise ($\diamond_k$ is an arbitrary element).

Is $M(g)$ well-defined?
Deciding Equality + (uninterpreted) Functions

Building the interpretation for function symbols

- $M(g)$ is a mapping from $|M|$ to $|M|$
- Defined as:

$$M(g)(\star_i) = \star_j \text{ if there is } v \equiv g(a) \text{ s.t.}$$

$$\begin{align*}
M(a) &= \star_i \\
M(v) &= \star_j
\end{align*}$$

$$= \star_k, \text{ otherwise (} \star_k \text{ is an arbitrary element)}$$

Is $M(g)$ well-defined? Problem: we may have

$v \equiv g(a)$ and $w \equiv g(b)$ s.t.

$M(a) = M(b) = \star_1$ and $M(v) = \star_2 \neq \star_3 = M(w)$

So, is $M(g)(\star_1) = \star_2$ or $M(g)(\star_1) = \star_3$?
Deciding Equality +
(uninterpreted) Functions

Building the interpretation for function symbols

- $M(g)$ is a mapping from $|M|$ to $|M|$
- Defined as:
  $$M(g)(i) = j \text{ if there is } v \equiv g(a) \text{ s.t.}$$
  $$M(a) = i,$$
  $$M(v) = j,$$
  $$= k, \text{ otherwise (}k\text{ is an arbitrary element)}.$$

Is $M(g)$ well-defined? Problem: we may have

$v \equiv g(a)$ and $w \equiv g(b)$ s.t.

$M(a) = M(b) = 1$ and $M(v) = 2 \neq 3 = M(w)$

So, is $M(g)(1) = 2$ or $M(g)(1) = 3$?
Deciding Equality + (uninterpreted) Functions

\[ a = b, \ b = c, \ d = e, \ b = s, \ d = t, \ a \neq v_4, \ v_2 \neq v_3 \]
\[ v_1 \equiv g(d), \ v_2 \equiv g(e), \ v_3 \equiv f(a, v_1), \ v_4 \equiv f(b, v_2) \]

Model construction:

\[ |M| = \{ \diamond_1, \diamond_2, \diamond_3, \diamond_4 \} \]
\[ M(a) = M(b) = M(c) = M(s) = \diamond_1 \]
\[ M(d) = M(e) = M(t) = \diamond_2 \]
\[ M(v_1) = M(v_2) = \diamond_3 \]
\[ M(v_3) = M(v_4) = \diamond_4 \]
Deciding Equality + (uninterpreted) Functions

\[ a = b, \ b = c, \ d = e, \ b = s, \ d = t, \ a \neq v_4, \ v_2 \neq v_3 \]

\[ v_1 \equiv g(d), \ v_2 \equiv g(e), \ v_3 \equiv f(a, v_1), \ v_4 \equiv f(b, v_2) \]

**Model construction:**

\[
|M| = \{\Diamond_1, \Diamond_2, \Diamond_3, \Diamond_4\}
\]

\[ M(a) = M(b) = M(c) = M(s) = \Diamond_1 \]

\[ M(d) = M(e) = M(t) = \Diamond_2 \]

\[ M(v_1) = M(v_2) = \Diamond_3 \]

\[ M(v_3) = M(v_4) = \Diamond_4 \]

\[ M(g)(\Diamond_i) = \Diamond_j \text{ if there is } v \equiv g(a) \text{ s.t.} \]

\[ M(a) = \Diamond_i \]

\[ M(v) = \Diamond_j \]

\[ = \Diamond_k, \text{ otherwise} \]
Deciding Equality + (uninterpreted) Functions

\[ a = b, \ b = c, \ d = e, \ b = s, \ d = t, \ a \neq v_4, \ v_2 \neq v_3 \]

\[ v_1 \equiv g(d), \ v_2 \equiv g(e), \ v_3 \equiv f(a, v_1), \ v_4 \equiv f(b, v_2) \]

Model construction:

\[ |M| = \{ \clubsuit_1, \clubsuit_2, \clubsuit_3, \clubsuit_4 \} \]

\[ M(a) = M(b) = M(c) = M(s) = \clubsuit_1 \]
\[ M(d) = M(e) = M(t) = \clubsuit_2 \]
\[ M(v_1) = M(v_2) = \clubsuit_3 \]
\[ M(v_3) = M(v_4) = \clubsuit_4 \]
\[ M(g) = \{ \clubsuit_2 \rightarrow \clubsuit_3 \} \]

\[ M(g)(\clubsuit_i) = \clubsuit_j \text{ if there is } v \equiv g(a) \text{ s.t.} \]
\[ M(a) = \clubsuit_i \]
\[ M(v) = \clubsuit_j \]
\[ = \clubsuit_k, \text{ otherwise} \]
Deciding Equality + (uninterpreted) Functions

\[ a = b, \quad b = c, \quad d = e, \quad b = s, \quad d = t, \quad a \neq v_4, \quad v_2 \neq v_3 \]

\[ v_1 \equiv g(d), \quad v_2 \equiv g(e), \quad v_3 \equiv f(a, \; v_1), \quad v_4 \equiv f(b, \; v_2) \]

Model construction:

\[ |M| = \{ \bullet_1, \bullet_2, \bullet_3, \bullet_4 \} \]

\[ M(a) = M(b) = M(c) = M(s) = \bullet_1 \]

\[ M(d) = M(e) = M(t) = \bullet_2 \]

\[ M(v_1) = M(v_2) = \bullet_3 \]

\[ M(v_3) = M(v_4) = \bullet_4 \]

\[ M(g) = \{ \bullet_2 \rightarrow \bullet_3 \} \]

\[ M(g)(\bullet_i) = \bullet_j \text{ if there is } v \equiv g(a) \text{ s.t.} \]

\[ M(a) = \bullet_i \]

\[ M(v) = \bullet_j \]

\[ = \bullet_k, \text{ otherwise} \]
Deciding Equality +
(uninterpreted) Functions

\[ a = b, \ b = c, \ d = e, \ b = s, \ d = t, \ a \neq v_4, \ v_2 \neq v_3 \]

\[ v_1 \equiv g(d), \ v_2 \equiv g(e), \ v_3 \equiv f(a, v_1), \ v_4 \equiv f(b, v_2) \]

**Model construction:**

\[ |M| = \{ \clubsuit_1, \clubsuit_2, \clubsuit_3, \clubsuit_4 \} \]

\[ M(a) = M(b) = M(c) = M(s) = \clubsuit_1 \]
\[ M(d) = M(e) = M(t) = \clubsuit_2 \]
\[ M(v_1) = M(v_2) = \clubsuit_3 \]
\[ M(v_3) = M(v_4) = \clubsuit_4 \]
\[ M(g) = \{ \clubsuit_2 \rightarrow \clubsuit_3, \text{else} \rightarrow \clubsuit_1 \} \]

\[ M(g)(\clubsuit_i) = \clubsuit_j \text{ if there is } v \equiv g(a) \text{ s.t.} \]
\[ M(a) = \clubsuit_i \]
\[ M(v) = \clubsuit_j \]
\[ = \clubsuit_k, \text{ otherwise} \]
Deciding Equality + (uninterpreted) Functions

\[ a = b, \ b = c, \ d = e, \ b = s, \ d = t, \ a \neq v_4, \ v_2 \neq v_3 \]

\[ v_1 \equiv g(d), \ v_2 \equiv g(e), \ v_3 \equiv f(a, v_1), \ v_4 \equiv f(b, v_2) \]

Model construction:

\[ |M| = \{ \spadesuit_1, \spadesuit_2, \spadesuit_3, \spadesuit_4 \} \]

\[ M(a) = M(b) = M(c) = M(s) = \spadesuit_1 \]

\[ M(d) = M(e) = M(t) = \spadesuit_2 \]

\[ M(v_1) = M(v_2) = \spadesuit_3 \]

\[ M(v_3) = M(v_4) = \spadesuit_4 \]

\[ M(g)(\spadesuit_i) = \spadesuit_j \text{ if there is } v \equiv g(a) \text{ s.t.} \]

\[ M(a) = \spadesuit_i \]

\[ M(v) = \spadesuit_j \]

\[ = \spadesuit_k, \text{ otherwise} \]

\[ M(f) = \{ (\spadesuit_1, \spadesuit_3) \rightarrow \spadesuit_4, \text{ else } \rightarrow \spadesuit_1 \} \]
Deciding Equality + (uninterpreted) Functions

What about predicates?

\[ p(a, b), \neg p(c, b) \]
Deciding Equality + (uninterpreted) Functions

What about predicates?

\[ p(a, b), \quad \neg p(c, b) \]

\[ f_p(a, b) = T, \quad f_p(c, b) \neq T \]
Deciding Equality + (uninterpreted) Functions

It is possible to implement our procedure in $O(n \log n)$.
Case Analysis

Many verification/analysis problems require:

\[ x \geq 0, \ y = x + 1, \ (y > 2 \lor y < 1) \]
Case Analysis

Many verification/analysis problems require:

\textbf{case-analysis}

\[
x \geq 0, \ y = x + 1, \ (y > 2 \vee y < 1)
\]

\textbf{Naïve Solution: Convert to DNF}

\[
(x \geq 0, \ y = x + 1, \ y > 2) \vee (x \geq 0, \ y = x + 1, \ y < 1)
\]
Case Analysis

Many verification/analysis problems require: 

\[
x \geq 0, \ y = x + 1, \ (y > 2 \lor y < 1)
\]

Naïve Solution: Convert to DNF

\[
(x \geq 0, \ y = x + 1, \ y > 2) \lor (x \geq 0, \ y = x + 1, \ y < 1)
\]

Too Inefficient!
(exponential blowup)
SMT : Basic Architecture

SAT + Theory Solvers = SMT

- Equality + UF
- Arithmetic
- Bit-vectors
- ...

Case Analysis
Guessing

\[ p \mid p \lor q, \neg q \lor r \]

\[ p, \neg q \mid p \lor q, \neg q \lor r \]
Deducing

\[
p \mid p \lor q, \neg p \lor s
\]

\[
p, s \mid p \lor q, \neg p \lor s
\]
Backtracking

\[ p, \neg s, q \mid p \lor q, s \lor q, \neg p \lor \neg q \]

\[ p, s \mid p \lor q, s \lor q, \neg p \lor \neg q \]
Basic Idea

\[ x \geq 0, \ y = x + 1, \ (y > 2 \lor y < 1) \]

Abstract (aka “naming” atoms)

\[ p_1, \ p_2, \ (p_3 \lor p_4) \]

\[ p_1 \equiv (x \geq 0), \ p_2 \equiv (y = x + 1), \]
\[ p_3 \equiv (y > 2), \ p_4 \equiv (y < 1) \]
Basic Idea

\[ x \geq 0, \; y = x + 1, \; (y > 2 \lor y < 1) \]

Abstract (aka “naming” atoms)

\[ p_1, \; p_2, \; (p_3 \lor p_4) \]

\[ p_1 \equiv (x \geq 0), \; p_2 \equiv (y = x + 1), \]
\[ p_3 \equiv (y > 2), \; p_4 \equiv (y < 1) \]
Basic Idea

\[ x \geq 0, \; y = x + 1, \; (y > 2 \lor y < 1) \]

Abstract (aka “naming” atoms)

\[ p_1, \; p_2, \; (p_3 \lor p_4) \]

\[ p_1 \equiv (x \geq 0), \; p_2 \equiv (y = x + 1), \]
\[ p_3 \equiv (y > 2), \; p_4 \equiv (y < 1) \]

Assignment

\[ p_1, \; p_2, \; \neg p_3, \; p_4 \]
Basic Idea

\[ x \geq 0, \ y = x + 1, \ (y > 2 \lor y < 1) \]

Abstract (aka “naming” atoms)

\[ p_1, \ p_2, \ (p_3 \lor p_4) \]

\[ p_1 \equiv (x \geq 0), \ p_2 \equiv (y = x + 1), \]
\[ p_3 \equiv (y > 2), \ p_4 \equiv (y < 1) \]

SAT Solver

Assignment

\[ p_1, \ p_2, \ \neg p_3, \ p_4 \]

\[ x \geq 0, \ y = x + 1, \]
\[ \neg (y > 2), \ y < 1 \]
Basic Idea

\[ x \geq 0, y = x + 1, (y > 2 \lor y < 1) \]

Abstract (aka “naming” atoms)

\[ p_1, p_2, (p_3 \lor p_4) \]

\[ p_1 \equiv (x \geq 0), p_2 \equiv (y = x + 1), p_3 \equiv (y > 2), p_4 \equiv (y < 1) \]

SAT Solver

Assignment

\[ p_1, p_2, \neg p_3, p_4 \]

\[ x \geq 0, y = x + 1, \neg(y > 2), y < 1 \]

Unsatisfiable

\[ x \geq 0, y = x + 1, y < 1 \]

Theory Solver
SAT + Theory solvers

Basic Idea

\[ x \geq 0, \ y = x + 1, \ (y > 2 \lor y < 1) \]

Abstract (aka "naming" atoms)

\[ p_1, \ p_2, \ (p_3 \lor p_4) \]
\[ p_1 \equiv (x \geq 0), \ p_2 \equiv (y = x + 1), \]
\[ p_3 \equiv (y > 2), \ p_4 \equiv (y < 1) \]

SAT Solver

Assignment

\[ p_1, \ p_2, \ \neg p_3, \ p_4 \]
\[ x \geq 0, \ y = x + 1, \]
\[ \neg (y > 2), \ y < 1 \]

New Lemma

\[ \neg p_1 \lor \neg p_2 \lor \neg p_4 \]

Unsatisfiable

\[ x \geq 0, \ y = x + 1, \ y < 1 \]

Theory Solver
SAT + Theory solvers

New Lemma
$\neg p_1 \lor \neg p_2 \lor \neg p_4$

Unsatisfiable
$x \geq 0, y = x + 1, y < 1$

AKA
Theory conflict

Theory Solver
procedure SmtSolver(F)
    \((F_p, M) := \text{Abstract}(F)\)

    loop
        \((R, A) := \text{SAT\_solver}(F_p)\)
        \(\text{if } R = \text{UNSAT then return } \text{UNSAT}\)
        \(S := \text{Concretize}(A, M)\)
        \((R, S') := \text{Theory\_solver}(S)\)
        \(\text{if } R = \text{SAT then return } \text{SAT}\)
        \(L := \text{New\_Lemma}(S', M)\)
        Add L to \(F_p\)
Basic Idea

\[ F: x \geq 0, y = x + 1, (y > 2 \lor y < 1) \]

Abstract (aka “naming” atoms)

\[ F_p: p_1, p_2, (p_3 \lor p_4) \]

\[ M: p_1 \equiv (x \geq 0), p_2 \equiv (y = x + 1), p_3 \equiv (y > 2), p_4 \equiv (y < 1) \]

\[ A: \text{Assignment} \]

\[ p_1, p_2, \neg p_3, p_4 \]

\[ S: x \geq 0, y = x + 1, \neg(y > 2), y < 1 \]

\[ L: \text{New Lemma} \]

\[ \neg p_1 \lor \neg p_2 \lor \neg p_4 \]

\[ S': \text{Unsatisfiable} \]

\[ x \geq 0, y = x + 1, y < 1 \]

\[ S': \text{Unsatisfiable} \]

\[ x \geq 0, y = x + 1, y < 1 \]
SAT + Theory solvers

**F**: \( x \geq 0, y = x + 1, (y > 2 \lor y < 1) \)

Abstract (aka “naming” atoms)

**F_p**: \( p_1, p_2, (p_3 \lor p_4) \)

**M**: \( p_1 \equiv (x \geq 0), p_2 \equiv (y = x + 1), p_3 \equiv (y > 2), p_4 \equiv (y < 1) \)

**A**: Assignment
\( p_1, p_2, \neg p_3, p_4 \)

**S**: \( x \geq 0, y = x + 1, \neg (y > 2), y < 1 \)

**L**: New Lemma
\( \neg p_1 \lor \neg p_2 \lor \neg p_4 \)

**S’**: Unsatisfiable
\( x \geq 0, y = x + 1, y < 1 \)

procedure SMT_Solver(F)

\((F_p, M) := \text{Abstract}(F)\)

loop

\((R, A) := \text{SAT Solver}(F_p)\)

if \( R = \text{UNSAT} \) then return UNSAT

\( S = \text{Concretize}(A, M) \)

\((R, S’) := \text{Theory Solver}(S)\)

if \( R = \text{SAT} \) then return SAT

\( L := \text{New Lemma}(S, M) \)

Add \( L \) to \( F_p \)

“Lazy translation” to DNF
State-of-the-art SMT solvers implement many improvements.
Incrementality
Send the literals to the Theory solver as they are assigned by the SAT solver

\[ p_1 \equiv (x \geq 0), \quad p_2 \equiv (y = x + 1), \]
\[ p_3 \equiv (y > 2), \quad p_4 \equiv (y < 1), \quad p_5 \equiv (x < 2), \]
\[ p_1, p_2, p_4 \mid p_1, p_2, (p_3 \lor p_4), (p_5 \lor \neg p_4) \]

Partial assignment is already Theory inconsistent.
Efficient Backtracking

We don’t want to restart from scratch after each backtracking operation.
Efficient Lemma Generation (computing a small $S'$)
Avoid lemmas containing redundant literals.

\[ p_1 \equiv (x \geq 0), \quad p_2 \equiv (y = x + 1), \]
\[ p_3 \equiv (y > 2), \quad p_4 \equiv (y < 1), \quad p_5 \equiv (x < 2), \]
\[ p_1, p_2, p_3, p_4 \mid p_1, p_2, (p_3 \lor p_4), (p_5 \lor \neg p_4) \]

\[ \neg p_1 \lor \neg p_2 \lor \neg p_3 \lor \neg p_4 \]

Imprecise Lemma
Theory Propagation

It is the SMT equivalent of unit propagation.

\[ p_1 \equiv (x \geq 0), \ p_2 \equiv (y = x + 1), \]
\[ p_3 \equiv (y > 2), \ p_4 \equiv (y < 1), \ p_5 \equiv (x < 2), \]
\[ p_1, \ p_2 \models p_1, \ p_2, (p_3 \lor p_4), (p_5 \lor \neg p_4) \]

\[ p_1, \ p_2 \ \text{imply} \ \neg p_4 \ \text{by theory propagation} \]

\[ p_1, \ p_2, \neg p_4 \models p_1, \ p_2, (p_3 \lor p_4), (p_5 \lor \neg p_4) \]
Theory Propagation

It is the SMT equivalent of unit propagation.

\[ p_1 \equiv (x \geq 0), \quad p_2 \equiv (y = x + 1), \]
\[ p_3 \equiv (y > 2), \quad p_4 \equiv (y < 1), \quad p_5 \equiv (x < 2), \]

\[ p_1, \; p_2 | \; p_1, \; p_2, (p_3 \lor p_4), (p_5 \lor \neg p_4) \]

\[ p_1, \; p_2 \text{ imply } \neg p_4 \text{ by theory propagation} \]

\[ p_1, \; p_2, \neg p_4 | \; p_1, \; p_2, (p_3 \lor p_4), (p_5 \lor \neg p_4) \]

**Tradeoff between precision \times performance.**
An Architecture: the core

Core

- Arithmetic
- Bit-Vectors
- Scalar Values

Equality
Uninterpreted Functions

SAT Solver

Case Analysis
An Architecture: the core

Core

- Arithmetic
- Bit-Vectors
- Scalar Values

Equality
Uninterpreted
Functions

SAT Solver

Blackboard:
equalities,
disequalities,
predicates
In practice, we need a combination of theories.

\[ b + 2 = c \text{ and } f(\text{read(write}(a,b,3), c-2)) \neq f(c-b+1) \]

A theory is a set (potentially infinite) of first-order sentences.

**Main questions:**

Is the union of two theories \( T_1 \cup T_2 \) consistent?

Given a solvers for \( T_1 \) and \( T_2 \), how can we build a solver for \( T_1 \cup T_2 \)?
A Combination History

Foundations

1979 Nelson, Oppen - Framework
1996 Tinelli & Harindi. N.O Fix
2000 Barrett et.al N.O + Rewriting
2002 Zarba & Manna. “Nice” Theories
2004 Ghilardi et.al. N.O. Generalized

Efficiency using rewriting

1984 Shostak. Theory solvers
1996 Cyrluk et.al Shostak Fix #1
1998 B. Shostak with Constraints
2001 Rueß & Shankar Shostak Fix #2
2004 Ranise et.al. N.O + Superposition

2001: Moskewicz et.al. Efficient DPLL made guessing cheap

2006 Bruttomesso et.al. Delayed Theory Combination

2007 de Moura & B. Model-based Theory Combination

… 2013 Jojanovich et.al. polite, shiny, etc.
Two theories are disjoint if they do not share function/constant and predicate symbols.

= is the only exception.

Example:
The theories of arithmetic and arrays are disjoint.

Arithmetic symbols: \{0, -1, 1, -2, 2, ..., +, -, *, >, <, \geq, \leq\}
Array symbols: \{ read, write \}
Purification

It is a different name for our “naming” subterms procedure.

\[ b + 2 = c, \quad f(\text{read}(\text{write}(a, b, 3), c - 2)) \neq f(c - b + 1) \]

\[ b + 2 = c, \quad v_6 \neq v_7 \]
\[ v_1 = 3, \quad v_2 = \text{write}(a, b, v_1), \quad v_3 = c - 2, \quad v_4 = \text{read}(v_2, v_3), \]
\[ v_5 = c - b + 1, \quad v_6 = f(v_4), \quad v_7 = f(v_5) \]
It is a different name for our “naming” subterms procedure.

\[ b + 2 = c, \ f(\text{read(write(a,b,3), c-2)}) \neq f(c-b+1) \]

\[ b + 2 = c, \ v_6 \neq v_7 \]
\[ v_1 \equiv 3, \ v_2 \equiv \text{write}(a, b, v_1), \ v_3 \equiv c-2, \ v_4 \equiv \text{read}(v_2, v_3), \]
\[ v_5 \equiv c-b+1, \ v_6 \equiv f(v_4), \ v_7 \equiv f(v_5) \]

\[ b + 2 = c, \ v_1 \equiv 3, \ v_3 \equiv c-2, \ v_5 \equiv c-b+1, \]
\[ v_2 \equiv \text{write}(a, b, v_1), \ v_4 \equiv \text{read}(v_2, v_3), \]
\[ v_6 \equiv f(v_4), \ v_7 \equiv f(v_5), \ v_6 \neq v_7 \]
A theory is stably infinite if every satisfiable QFF is satisfiable in an infinite model.

EUF and arithmetic are stably infinite.

Bit-vectors are not.
The union of two consistent, disjoint, stably infinite theories is consistent.
A theory $T$ is **convex** iff

for all finite sets $S$ of literals and

for all $a_1 = b_1 \lor \ldots \lor a_n = b_n$

$S$ implies $a_1 = b_1 \lor \ldots \lor a_n = b_n$

iff

$S$ implies $a_i = b_i$ for some $1 \leq i \leq n$
Every convex theory with non trivial models is stably infinite.

All Horn equational theories are convex.

formulas of the form \( s_1 \neq r_1 \lor \ldots \lor s_n \neq r_n \lor t = t' \)

Linear rational arithmetic is convex.
Linear integer arithmetic is not convex
\[ 1 \leq a \leq 2, \ b = 1, \ c = 2 \text{ implies } a = b \lor a = c \]

Nonlinear arithmetic
\[ a^2 = 1, \ b = 1, \ c = -1 \text{ implies } a = b \lor a = c \]

Theory of bit-vectors

Theory of arrays
\[ c_1 = \text{read}(\text{write}(a, \ i, \ c_2), \ j), \ c_3 = \text{read}(a, \ j) \]
implies \[ c_1 = c_2 \lor c_1 = c_3 \]
EUF is convex (O(n log n))
IDL is non-convex (O(nm))

EUF ∪ IDL is NP-Complete

Reduce 3CNF to EUF ∪ IDL
For each boolean variable $p_i$ add $0 \leq a_i \leq 1$
For each clause $p_1 \lor \neg p_2 \lor p_3$ add
\[ f(a_1, a_2, a_3) \neq f(0, 1, 0) \]
EUF is convex (O(n log n))
IDL is non-convex (O(nm))

EUF \cup IDL is NP-Complete

Reduce 3CNF to EUF \cup IDL

For each boolean variable \( p_i \) add \( 0 \leq a_i \leq 1 \)
For each clause \( p_1 \lor \neg p_2 \lor p_3 \) add
\[ f(a_1, a_2, a_3) \neq f(0, 1, 0) \]

implies
\[ a_1 \neq 0 \lor a_2 \neq 1 \lor a_3 \neq 0 \]
Let $T_1$ and $T_2$ be consistent, stably infinite theories over disjoint (countable) signatures. Assume satisfiability of conjunction of literals can decided in $O(T_1(n))$ and $O(T_2(n))$ time respectively. Then,

1. The combined theory $T$ is consistent and stably infinite.

2. Satisfiability of quantifier free conjunction of literals in $T$ can be decided in $O(2^{n^2} \times (T_1(n) + T_2(n)))$.

3. If $T_1$ and $T_2$ are convex, then so is $T$ and satisfiability in $T$ is in $O(n^3 \times (T_1(n) + T_2(n)))$. 
The combination procedure:

**Initial State:** $\phi$ is a conjunction of literals over $\Sigma_1 \cup \Sigma_2$.

**Purification:** Preserving satisfiability transform $\phi$ into $\phi_1 \land \phi_2$, such that, $\phi_i \in \Sigma_i$.

**Interaction:** Guess a partition of $\mathcal{V}(\phi_1) \cap \mathcal{V}(\phi_2)$ into disjoint subsets. Express it as conjunction of literals $\psi$.

Example. The partition $\{x_1\}, \{x_2, x_3\}, \{x_4\}$ is represented as $x_1 \neq x_2, x_1 \neq x_4, x_2 \neq x_4, x_2 = x_3$.

**Component Procedures:** Use individual procedures to decide whether $\phi_i \land \psi$ is satisfiable.

**Return:** If both return yes, return yes. No, otherwise.
Instead of guessing, we can deduce the equalities to be shared.

**Purification:** no changes.

**Interaction:** Deduce an equality $x = y$:

$$\mathcal{T}_1 \vdash (\phi_1 \Rightarrow x = y)$$

Update $\phi_2 := \phi_2 \land x = y$. And vice-versa. Repeat until no further changes.

**Component Procedures** : Use individual procedures to decide whether $\phi_i$ is satisfiable.

Remark: $\mathcal{T}_i \vdash (\phi_i \Rightarrow x = y)$ iff $\phi_i \land x \neq y$ is not satisfiable in $\mathcal{T}_i$. 
NO deterministic procedure
Completeness

Assume the theories are convex.

- Suppose $\phi_i$ is satisfiable.
- Let $E$ be the set of equalities $x_j = x_k$ ($j \neq k$) such that,
  $\mathcal{T}_i \not\models \phi_i \Rightarrow x_j = x_k$.
- By convexity, $\mathcal{T}_i \not\models \phi_i \Rightarrow \bigvee_E x_j = x_k$.
- $\phi_i \land \bigwedge_E x_j \neq x_k$ is satisfiable.
- The proof now is identical to the nondeterministic case.
- Sharing equalities is sufficient, because a theory $\mathcal{T}_1$ can assume that $x^B \neq y^B$ whenever $x = y$ is not implied by $\mathcal{T}_2$ and vice versa.
\[ b + 2 = c, \quad f(\text{read(write(a,b,3), c-2)}) \neq f(c-b+1) \]

**Arithmetic**
- \[ b + 2 = c, \]
- \[ v_1 \equiv 3, \]
- \[ v_3 \equiv c-2, \]
- \[ v_5 \equiv c-b+1 \]

**Arrays**
- \[ v_2 \equiv \text{write(a, b, } v_1), \]
- \[ v_4 \equiv \text{read}(v_2, v_3) \]

**EUF**
- \[ v_6 \equiv f(v_4), \]
- \[ v_7 \equiv f(v_5), \]
- \[ v_6 \neq v_7 \]
b + 2 = c, f(read(write(a,b,3), c-2)) ≠ f(c-b+1)

Arithmetic
b + 2 = c,
v_1 ≡ 3,
v_3 ≡ c-2,
v_5 ≡ c-b+1

Arrays
v_2 ≡ write(a, b, v_1),
v_4 ≡ read(v_2, v_3)

EUF
v_6 ≡ f(v_4),
v_7 ≡ f(v_5),
v_6 ≠ v_7

Substituting c
b + 2 = c, \( f(\text{read}(\text{write}(a, b, 3), c-2)) \neq f(c-b+1) \)

**Arithmetic**

\[b + 2 = c, \quad v_1 = 3, \quad v_3 = b, \quad v_5 = 3\]

**Arrays**

\[v_2 \equiv \text{write}(a, b, v_1), \quad v_4 \equiv \text{read}(v_2, v_3)\]

**EUF**

\[v_6 \equiv f(v_4), \quad v_7 \equiv f(v_5), \quad v_6 \neq v_7\]

**Propagating** \( v_3 = b \)
**Arithmetic**

\[ b + 2 = c, \quad f(\text{read}(\text{write}(a, b, 3), c - 2)) \neq f(c - b + 1) \]

**Arrays**

\[ v_2 \equiv \text{write}(a, b, v_1), \quad v_4 \equiv \text{read}(v_2, v_3), \quad v_3 = b \]

**EUF**

\[ v_6 \equiv f(v_4), \quad v_7 \equiv f(v_5), \quad v_6 \neq v_7, \quad v_3 = b \]

Deducing \( v_4 = v_1 \)
b + 2 = c, \( f(\text{read}(\text{write}(a, b, 3), c-2)) \neq f(c-b+1) \)

### Arithmetic
- \( b + 2 = c \),
- \( v_1 \equiv 3 \),
- \( v_3 \equiv b \),
- \( v_5 \equiv 3 \)

### Arrays
- \( v_2 \equiv \text{write}(a, b, v_1) \),
- \( v_4 \equiv \text{read}(v_2, v_3) \),
- \( v_3 = b \),
- \( v_4 = v_1 \)

### EUF
- \( v_6 \equiv f(v_4) \),
- \( v_7 \equiv f(v_5) \),
- \( v_6 \neq v_7 \),
- \( v_3 = b \)

Propagating \( v_4 = v_1 \)
b + 2 = c, f(read(write(a, b, 3), c-2)) ≠ f(c-b+1)

Arithmetic
b + 2 = c,
v₁ ≡ 3,
v₃ ≡ b,
v₅ ≡ 3,
v₄ = v₁

Arrays
v₂ ≡ write(a, b, v₁),
v₄ ≡ read(v₂, v₃),
v₃ = b,
v₄ = v₁

EUF
v₆ ≡ f(v₄),
v₇ ≡ f(v₅),
v₆ ≠ v₇,
v₃ = b,
v₄ = v₁

Propagating v₅ = v₁
NO procedure: Example

\[ b + 2 = c, \quad f(\text{read(write(a,b,3), c-2)}) \neq f(c-b+1) \]

**Arithmetic**

\[ b + 2 = c, \quad v_1 \equiv 3, \quad v_3 = b, \quad v_5 \equiv 3, \quad v_4 = v_1 \]

**Arrays**

\[ v_2 \equiv \text{write}(a, b, v_1), \quad v_4 \equiv \text{read}(v_2, v_3), \quad v_3 = b, \quad v_4 = v_1 \]

**EUF**

\[ v_6 \equiv f(v_4), \quad v_7 \equiv f(v_5), \quad v_6 \neq v_7, \quad v_3 = b, \quad v_4 = v_1, \quad v_5 = v_1 \]

**Congruence:** \[ v_6 = v_7 \]
**NO procedure: Example**

\[ b + 2 = c, \quad f(\text{read}(\text{write}(a, b, 3), c-2)) \neq f(c-b+1) \]

<table>
<thead>
<tr>
<th>Arithmetic</th>
<th>Arrays</th>
<th>EUF</th>
</tr>
</thead>
<tbody>
<tr>
<td>[ b + 2 = c, ]</td>
<td>[ v_2 \equiv \text{write}(a, b, v_1), ]</td>
<td>[ v_6 \equiv f(v_4), ]</td>
</tr>
<tr>
<td>[ v_1 \equiv 3, ]</td>
<td>[ v_4 \equiv \text{read}(v_2, v_3), ]</td>
<td>[ v_7 \equiv f(v_5), ]</td>
</tr>
<tr>
<td>[ v_3 \equiv b, ]</td>
<td>[ v_3 = b, ]</td>
<td>[ v_6 \neq v_7, ]</td>
</tr>
<tr>
<td>[ v_5 \equiv 3, ]</td>
<td>[ v_4 = v_1 ]</td>
<td>[ v_3 = b, ]</td>
</tr>
<tr>
<td>[ v_4 = v_1 ]</td>
<td></td>
<td>[ v_4 = v_1, ]</td>
</tr>
</tbody>
</table>

**Unsatisfiable**
Deterministic procedure may fail for non-convex theories.

\[ 0 \leq a \leq 1, \ 0 \leq b \leq 1, \ 0 \leq c \leq 1, \]
\[ f(a) \neq f(b), \]
\[ f(a) \neq f(c), \]
\[ f(b) \neq f(c) \]
Combining Procedures in Practice

- Propagate all implied equalities.
  - Deterministic Nelson-Oppen.
  - Complete only for convex theories.
  - It may be expensive for some theories.

Delayed Theory Combination.

- Nondeterministic Nelson-Oppen.
- Create set of interface equalities \((x = y)\) between shared variables.
- Use SAT solver to guess the partition.
- Disadvantage: the number of additional equality literals is quadratic in the number of shared variables.
Common to these methods is that they are pessimistic about which equalities are propagated.

Model-based Theory Combination

- Optimistic approach.
- Use a candidate model $M_i$ for one of the theories $T_i$ and propagate all equalities implied by the candidate model, hedging that other theories will agree.

\[
\text{if } M_i \models T_i \cup \Gamma_i \cup \{u = v\} \text{ then propagate } u = v.
\]
- If not, use backtracking to fix the model.
- It is cheaper to enumerate equalities that are implied in a particular model than of all models.
\[ x = f(y - 1), \quad f(x) \neq f(y), \quad 0 \leq x \leq 1, \quad 0 \leq y \leq 1 \]

Purifying
Example

\[ x = f(z), \, f(x) \neq f(y), \, 0 \leq x \leq 1, \, 0 \leq y \leq 1, \, z = y - 1 \]
### Example

<table>
<thead>
<tr>
<th>$\mathcal{T}_E$</th>
<th>$\mathcal{T}_A$</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Literals</strong></td>
<td><strong>Eq. Classes</strong></td>
</tr>
<tr>
<td>$x = f(z)$</td>
<td>${x, f(z)}$</td>
</tr>
<tr>
<td>$f(x) \neq f(y)$</td>
<td>${y}$</td>
</tr>
<tr>
<td></td>
<td>${z}$</td>
</tr>
<tr>
<td></td>
<td>${f(x)}$</td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Assume $x = y$
### Example

<table>
<thead>
<tr>
<th>$\mathcal{T}_E$</th>
<th>$\mathcal{T}_A$</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Literals</strong></td>
<td><strong>Model</strong></td>
</tr>
<tr>
<td>$x = f(z)$</td>
<td>$E(x) = *_{1}$</td>
</tr>
<tr>
<td>$f(x) \neq f(y)$</td>
<td>$E(y) = *_{1}$</td>
</tr>
<tr>
<td>$x = y$</td>
<td>$E(z) = *_{2}$</td>
</tr>
<tr>
<td></td>
<td>$E(f) = {*<em>{1} \mapsto *</em>{3},$</td>
</tr>
<tr>
<td></td>
<td>$*<em>{2} \mapsto *</em>{1},$</td>
</tr>
<tr>
<td></td>
<td>$else \mapsto *_{4}}$</td>
</tr>
</tbody>
</table>

Unsatisfiable
### \( \mathcal{T}_E \) and \( \mathcal{T}_A \)

<table>
<thead>
<tr>
<th>Literals</th>
<th>Eq. Classes</th>
<th>Model</th>
<th>Literals</th>
<th>Model</th>
</tr>
</thead>
<tbody>
<tr>
<td>( x = f(z) )</td>
<td>( {x, f(z)} )</td>
<td>( E(x) = *_1 )</td>
<td>( 0 \leq x \leq 1 )</td>
<td>( A(x) = 0 )</td>
</tr>
<tr>
<td>( f(x) \neq f(y) )</td>
<td>( {y} )</td>
<td>( E(y) = *_2 )</td>
<td>( 0 \leq y \leq 1 )</td>
<td>( A(y) = 0 )</td>
</tr>
<tr>
<td>( x \neq y )</td>
<td>( {z} )</td>
<td>( E(z) = *_3 )</td>
<td>( z = y - 1 )</td>
<td>( A(z) = -1 )</td>
</tr>
<tr>
<td>( {f(x)} )</td>
<td>( {f(y)} )</td>
<td>( E(f) = {*_1 \mapsto *_4, ) ( *_2 \mapsto *_5, ) ( *_3 \mapsto *_1, ) ( else \mapsto *_6} )</td>
<td>( x \neq y )</td>
<td></td>
</tr>
</tbody>
</table>

Backtrack, and assert \( x \neq y \).

\( \mathcal{T}_A \) model need to be fixed.
### Example

<table>
<thead>
<tr>
<th>( \mathcal{T}_E )</th>
<th>( \mathcal{T}_A )</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Liters</strong></td>
<td><strong>Eq. Classes</strong></td>
</tr>
<tr>
<td>( x = f(z) )</td>
<td>{( x, f(z) )}</td>
</tr>
<tr>
<td>( f(x) \neq f(y) )</td>
<td>{( y )}</td>
</tr>
<tr>
<td>( x \neq y )</td>
<td>{( z )}</td>
</tr>
<tr>
<td>{( f(x) )}</td>
<td>{( f(y) )}</td>
</tr>
</tbody>
</table>

Assume \( x = z \)
### Example

<table>
<thead>
<tr>
<th>( T_E )</th>
<th>( T_A )</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Literals</strong></td>
<td><strong>Eq. Classes</strong></td>
</tr>
<tr>
<td>( x = f(z) )</td>
<td>( {x, z, f(x), f(z)} )</td>
</tr>
<tr>
<td>( f(x) \neq f(y) )</td>
<td>( {y} )</td>
</tr>
<tr>
<td>( x \neq y )</td>
<td>( {f(y)} )</td>
</tr>
<tr>
<td>( x = z )</td>
<td></td>
</tr>
</tbody>
</table>

Satisfiable
### Example

<table>
<thead>
<tr>
<th>$T_E$</th>
<th>$T_A$</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Literals</strong></td>
<td><strong>Eq. Classes</strong></td>
</tr>
<tr>
<td>$x = f(z)$</td>
<td>${x, z, f(x), f(z)}$</td>
</tr>
<tr>
<td>$f(x) \neq f(y)$</td>
<td>${y}$</td>
</tr>
<tr>
<td>$x \neq y$</td>
<td>${f(y)}$</td>
</tr>
<tr>
<td>$x = z$</td>
<td></td>
</tr>
</tbody>
</table>

Let $h$ be the bijection between $|E|$ and $|A|$.

$$h = \{*_1 \mapsto 0, *_2 \mapsto 1, *_3 \mapsto -1, *_4 \mapsto 2, \ldots\$$
### Example

<table>
<thead>
<tr>
<th>$T_E$</th>
<th>$T_A$</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Literals</strong></td>
<td><strong>Model</strong></td>
</tr>
<tr>
<td>$x = f(z)$</td>
<td>$E(x) = *_1$</td>
</tr>
<tr>
<td>$f(x) \neq f(y)$</td>
<td>$E(y) = *_2$</td>
</tr>
<tr>
<td>$x \neq y$</td>
<td>$E(z) = *_1$</td>
</tr>
<tr>
<td>$x = z$</td>
<td>$E(f) = {*_1 \leftrightarrow <em>_1$, $</em>_2 \leftrightarrow *_3$, $\text{else} \leftrightarrow *_4}$</td>
</tr>
</tbody>
</table>

Extending $A$ using $h$.

$$h = \{*_1 \leftrightarrow 0, *_2 \leftrightarrow 1, *_3 \leftrightarrow -1, *_4 \leftrightarrow 2, \ldots\}$$