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

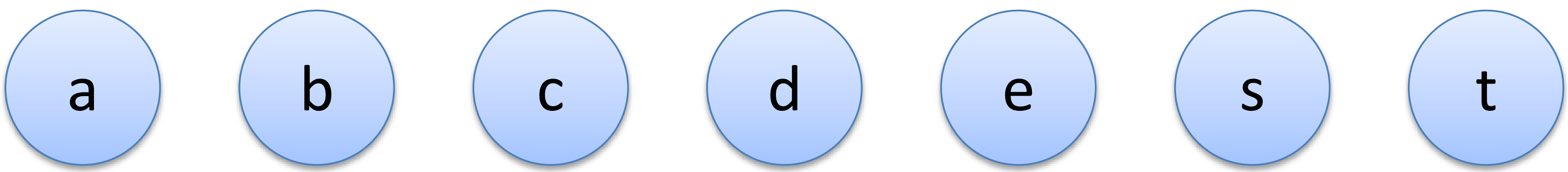
Fri Programming Z3: Interfacing and Solving

Lecture Overview

- Deciding Equality
- Uninterpreted Functions
- Nelson Oppen Combination
- Model-based Theory Combination

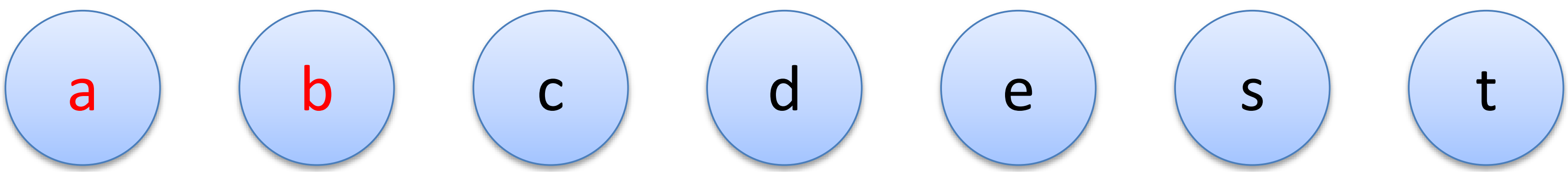
Deciding Equality

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



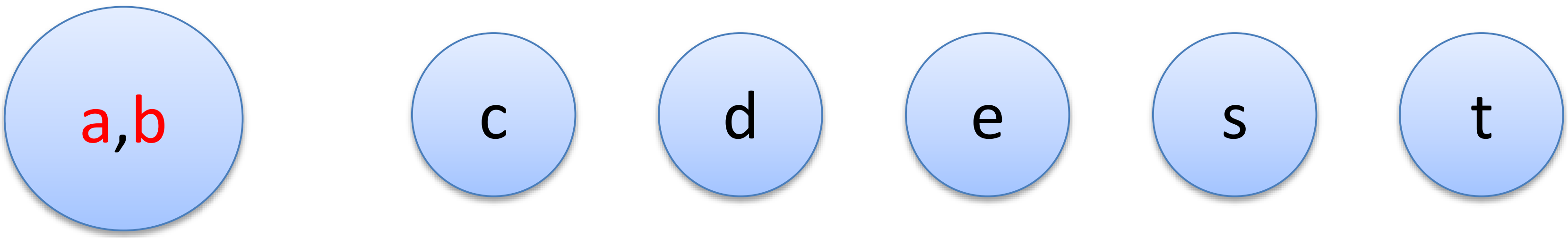
Deciding Equality

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



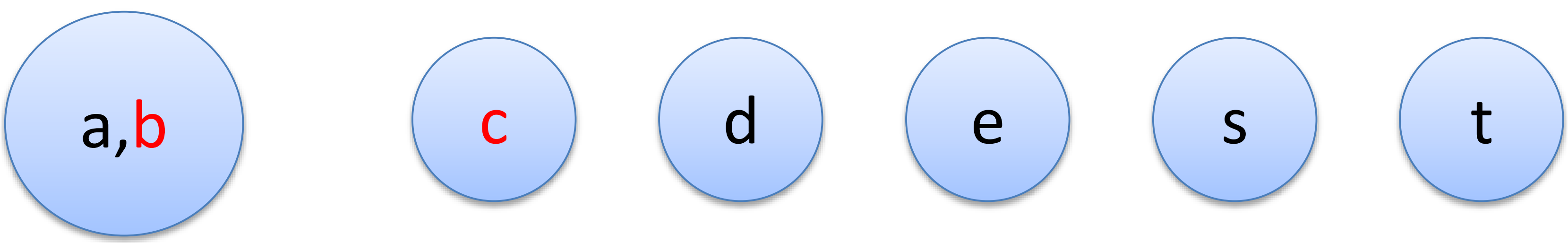
Deciding Equality

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



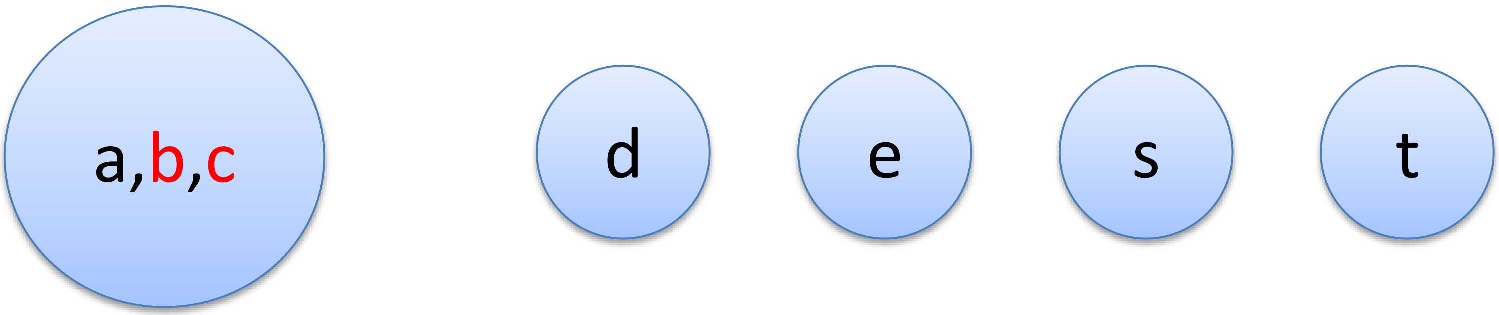
Deciding Equality

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



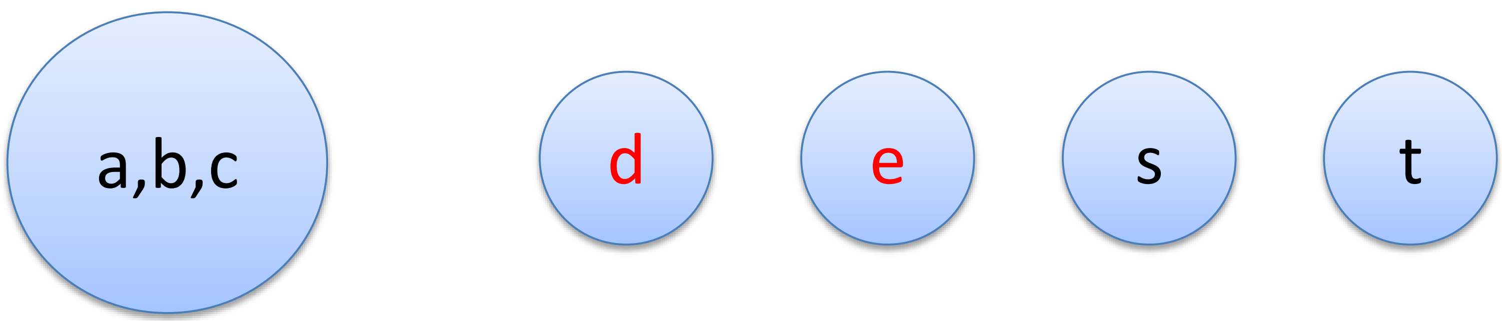
Deciding Equality

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



Deciding Equality

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

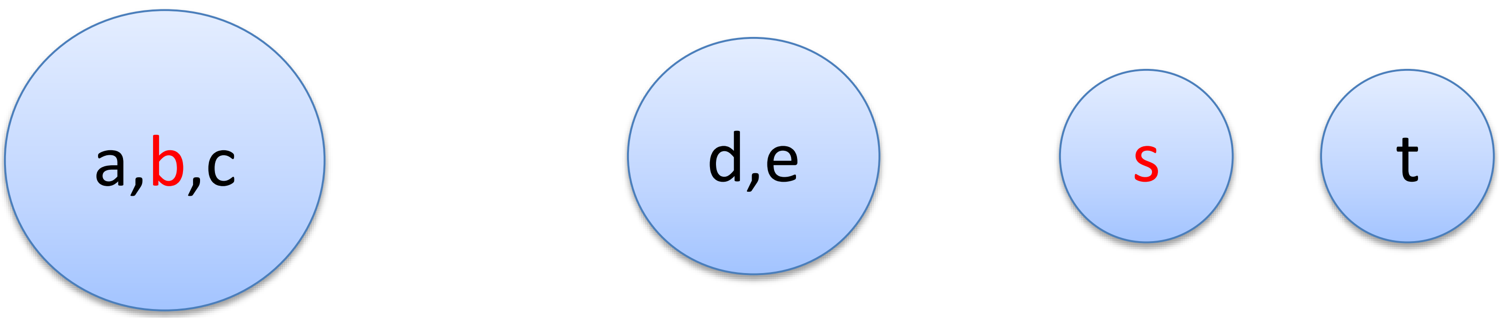


Deciding Equality

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

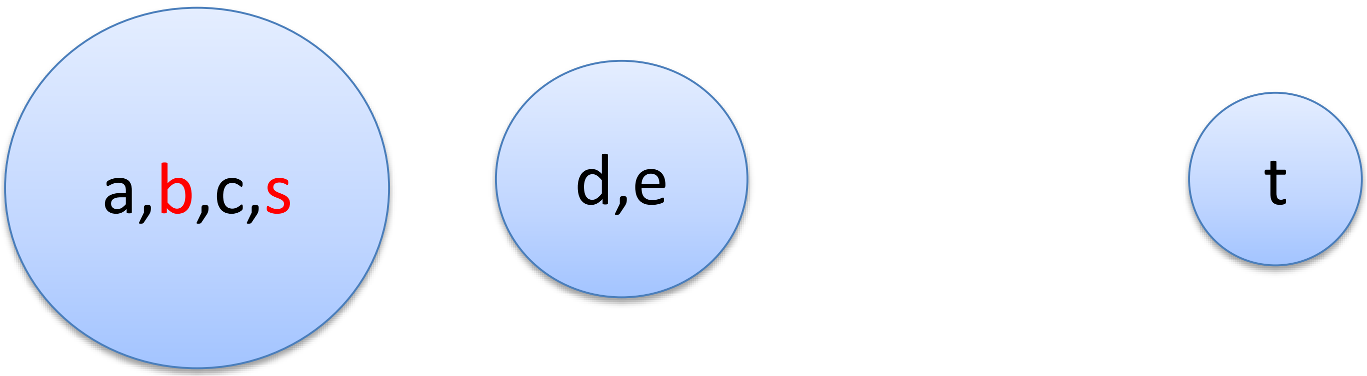
Deciding Equality

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



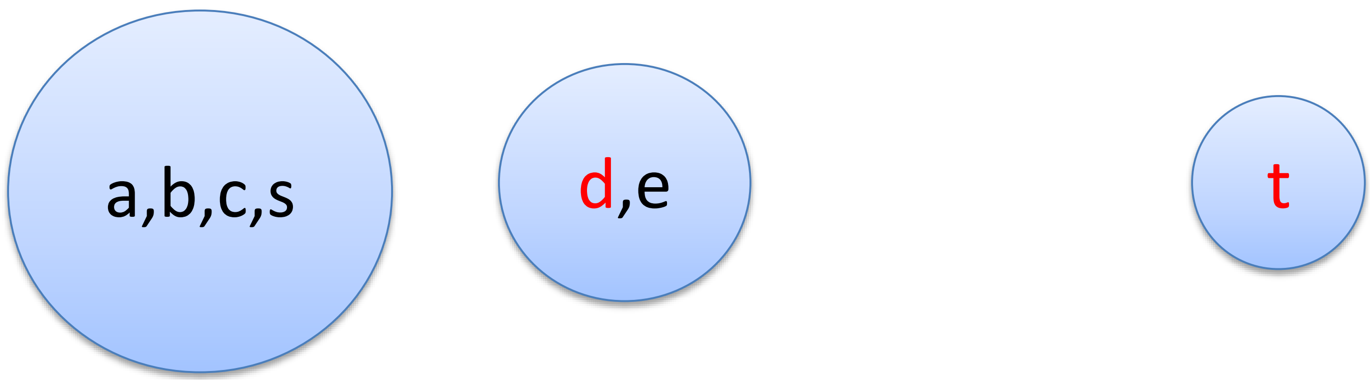
Deciding Equality

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



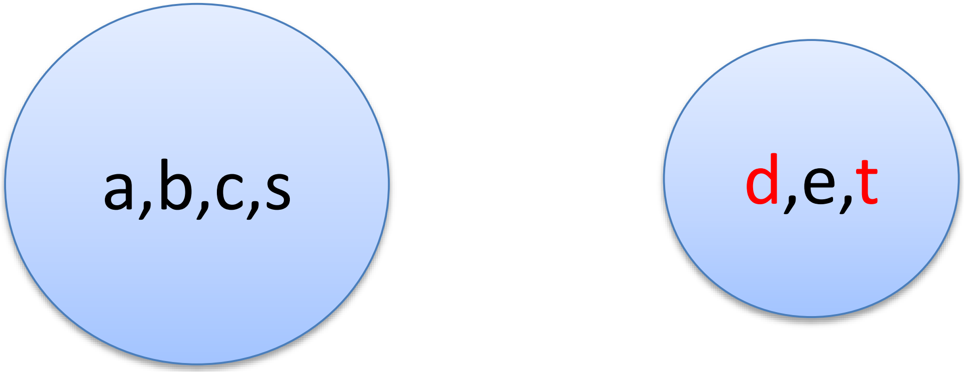
Deciding Equality

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



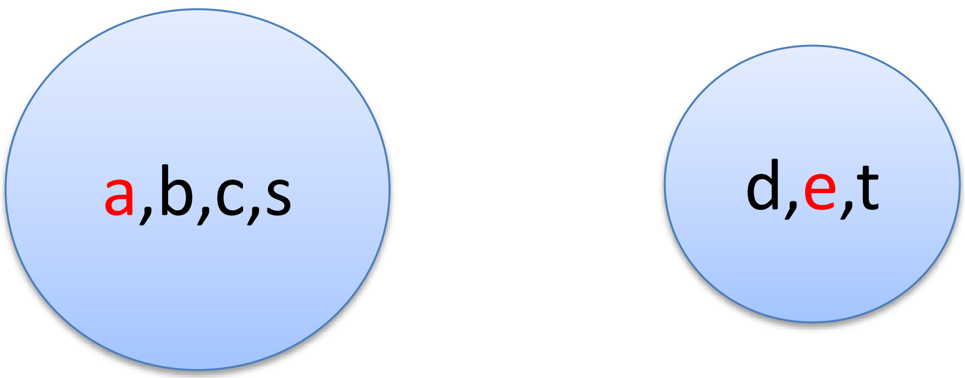
Deciding Equality

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



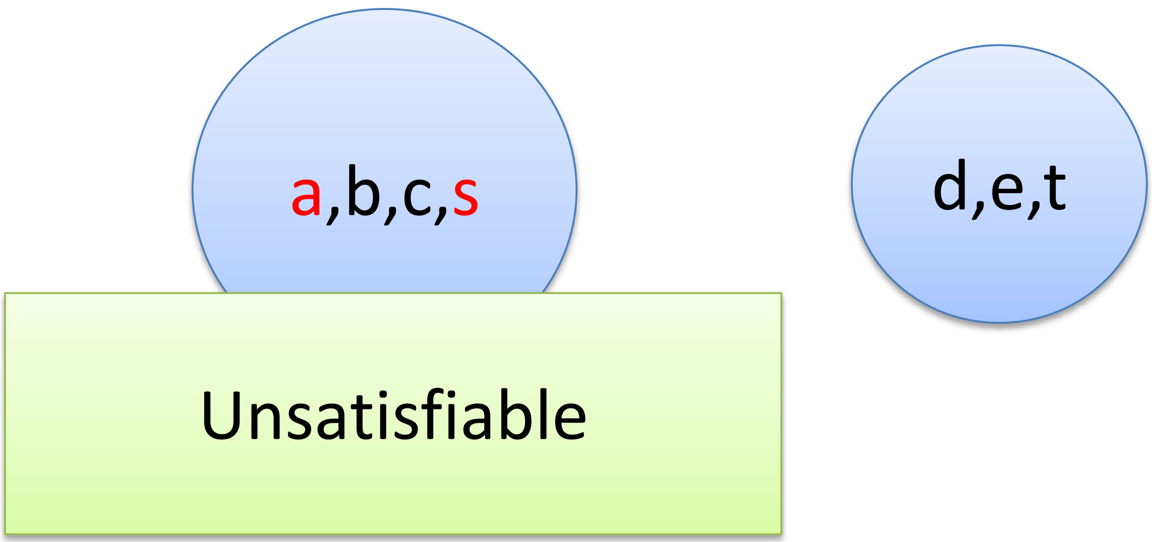
Deciding Equality

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



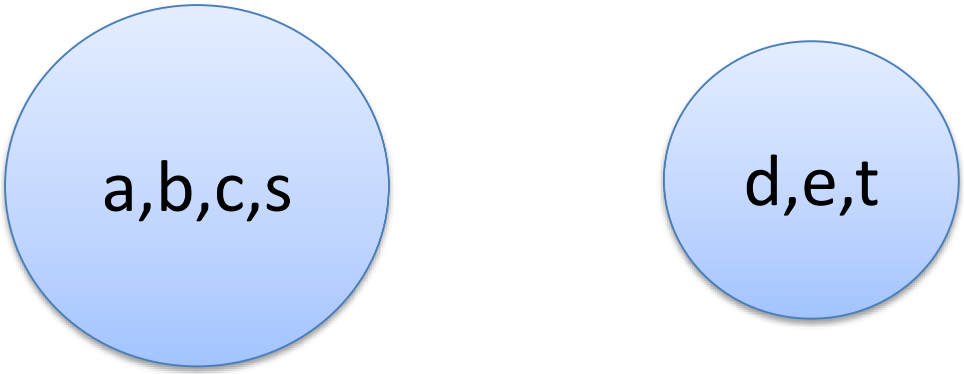
Deciding Equality

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



Deciding Equality

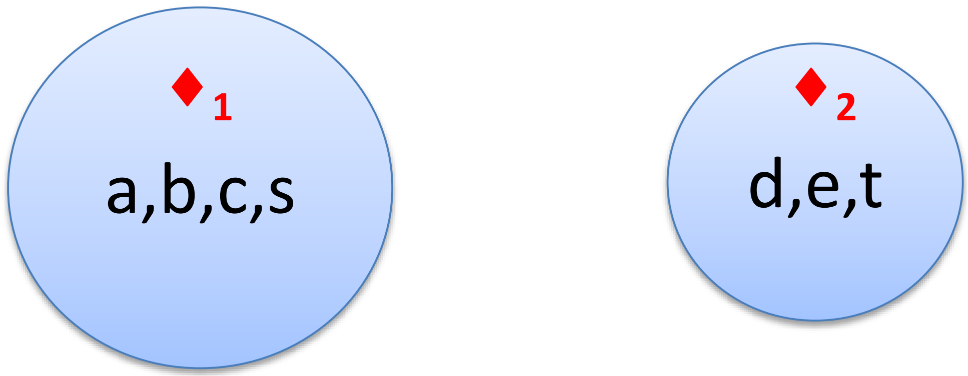
$a = b,\ b = c,\ d = e,\ b = s,\ d = t,\ a \neq e$



Model construction

Deciding Equality

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

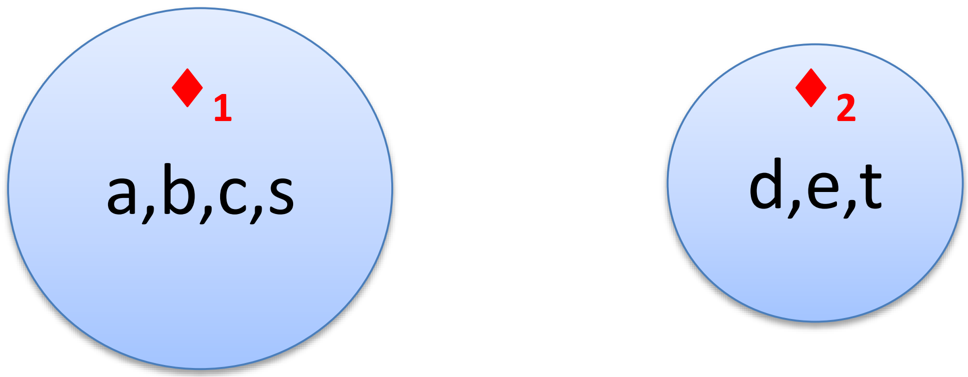


Model construction

$|M| = \{\blacklozenge_1, \blacklozenge_2\}$ (universe, aka domain)

Deciding Equality

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



Model construction

$|M| = \{\blacklozenge_1, \blacklozenge_2\}$ (universe, aka domain)

$M(a) = \blacklozenge_1$ (assignment)

Deciding Equality

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

Alternative notation:
$a^M = ♦_1$

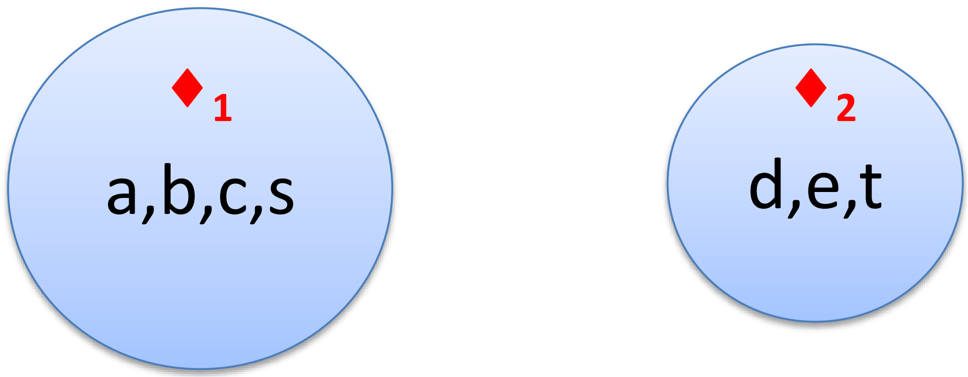
Model construction

$|M| = \{♦_1, ♦_2\}$ (universe, aka domain)

$M(a) = ♦_1$ (assignment)

Deciding Equality

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



Model construction

$|M| = \{\blacklozenge_1, \blacklozenge_2\}$ (universe, aka domain)

$M(a) = M(b) = M(c) = M(s) = \blacklozenge_1$

$M(d) = M(e) = M(t) = \blacklozenge_2$

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.

Equality: Union-Find [Tarjan]

```
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

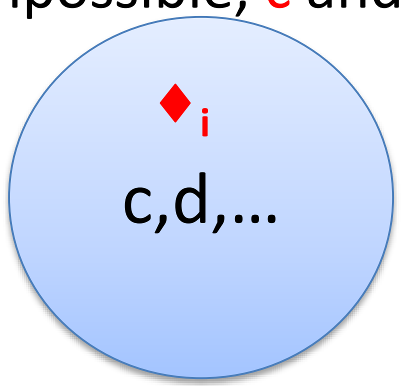
Lazy path compression Variant: Eager Path compression + equivalence class as doubly linked list

nlog*(n) amortized time for n operations

Root for largest class takes over

Deciding Equality: Termination, Soundness, Completeness

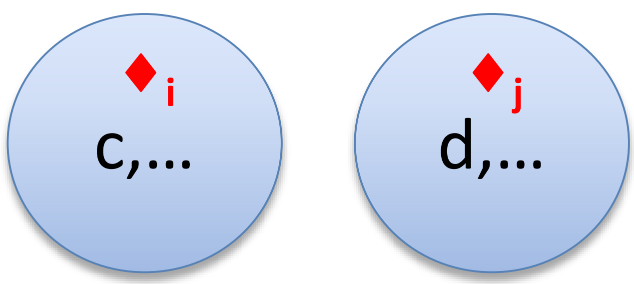
- Completeness
  - We can build a model if an inconsistency was not detected.
  - Instantiating the template for our procedure:
    - Assume some literal $c = d$ is not satisfied by our model.
    - That is, $M(c) \neq M(d)$.
    - This is impossible, c and d must be in the same “ball”.



$M(c) = M(d) = \blacklozenge_i$

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 $c \neq d$ 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”



$$M(c) = \blacklozenge_i$$
$$M(d) = \blacklozenge_j$$

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 $f(x_1, \ldots, x_n) = f(y_1, \ldots, y_n)$

Microsoft Research

Deciding Equality + (uninterpreted) Functions

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

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:

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

Microsoft Research

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:

$x_1 = y_1, \ldots, x_n = y_n$ 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:

$x_1 = y_1, \ldots, x_n = y_n$ implies $f(x_1, \ldots, x_n) = f(y_1, \ldots, y_n)$

Microsoft Research

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:

$x_1 = y_1, \ldots, x_n = y_n$ 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:

$x_1 = y_1, \ldots, x_n = y_n$ implies $f(x_1, \ldots, x_n) = f(y_1, \ldots, y_n)$

Deciding Equality + (uninterpreted) Functions

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

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

First Step: "Naming" subterms

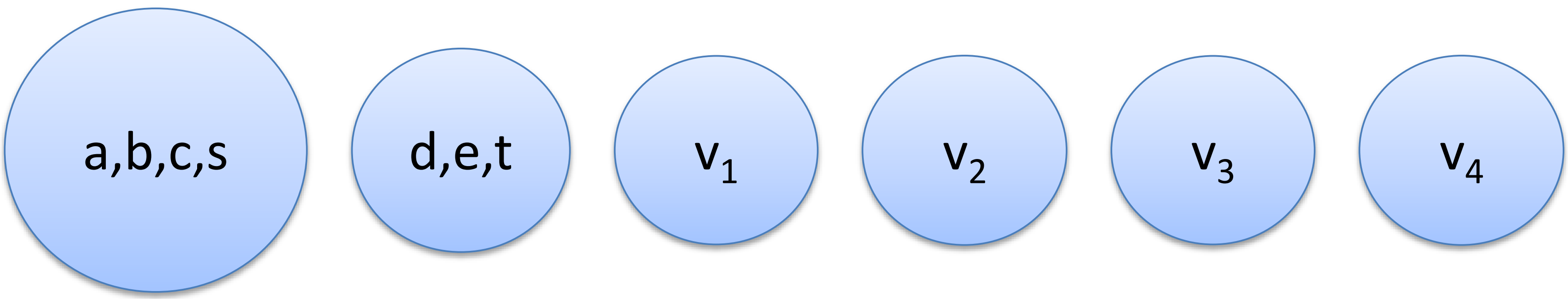
Congruence Rule:

$x_1 = y_1, \ldots, x_n = y_n$ implies $f(x_1, \ldots, x_n) = f(y_1, \ldots, y_n)$

Deciding Equality + (uninterpreted) Functions

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

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



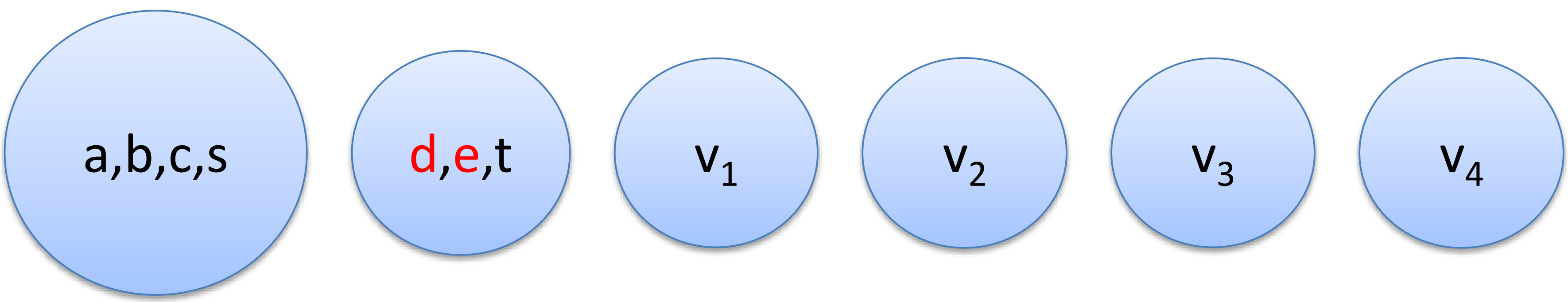
Congruence Rule:

$x_1 = y_1, \ldots, x_n = y_n$ 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:

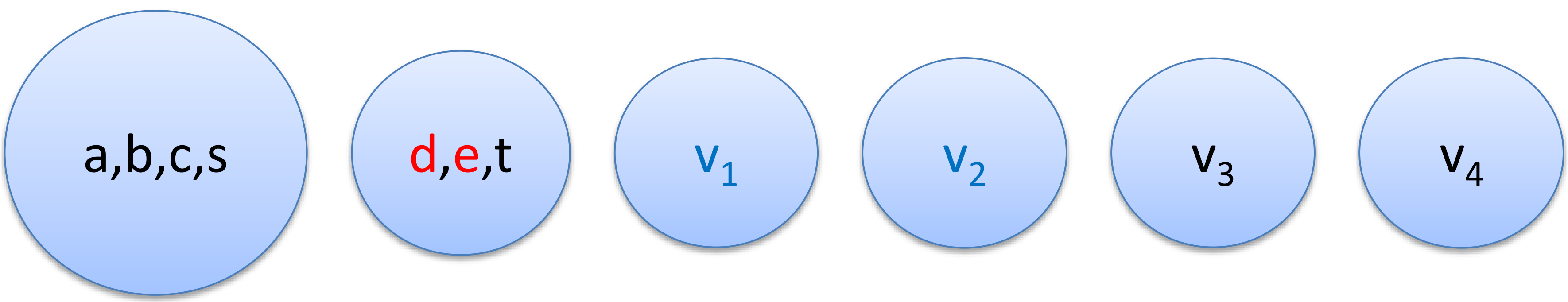
$x_1 = y_1, \ldots, x_n = y_n$ implies $f(x_1, \ldots, x_n) = f(y_1, \ldots, y_n)$

$d = e$ implies $g(d) = g(e)$

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 $f(x_1, \ldots, x_n) = f(y_1, \ldots, y_n)$

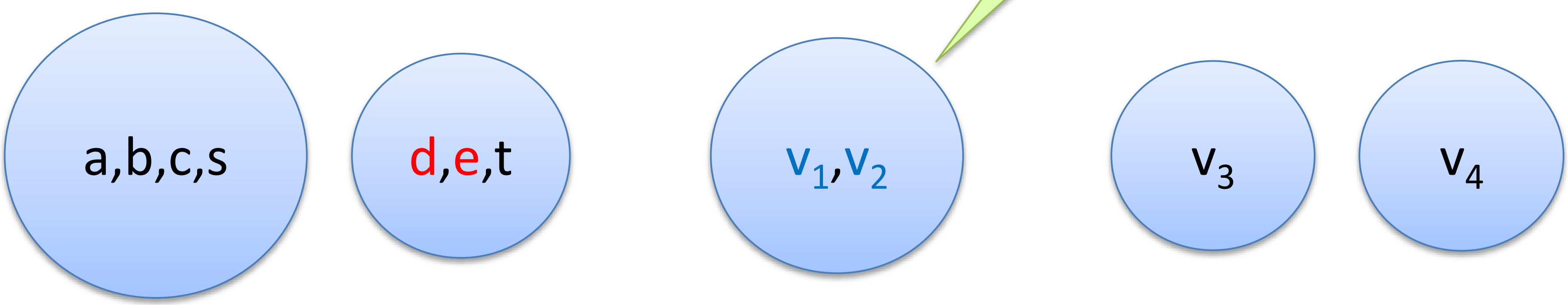
$d = e$ implies $v_1 = v_2$

Deciding Equality (uninterpreted) Funct

We say:
v_1 and v_2 are congruent.

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

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



Congruence Rule:

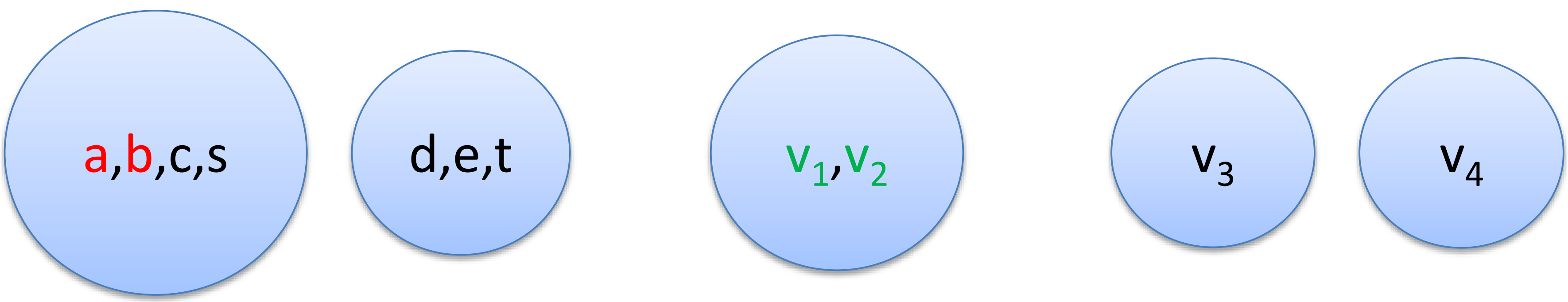
$x_1 = y_1, \ldots, x_n = y_n$ implies $f(x_1, \ldots, x_n) = f(y_1, \ldots, y_n)$

$d = e$ 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:

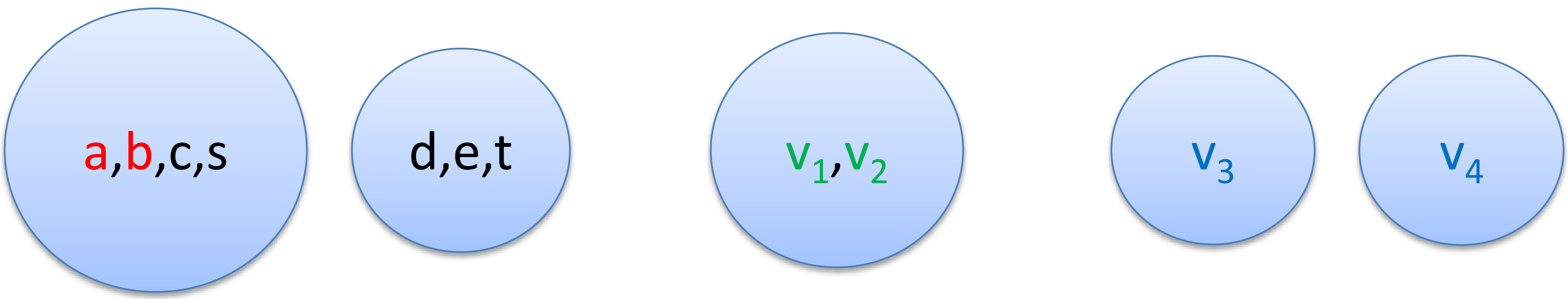
$$x_1 = y_1, \ldots, x_n = y_n \text{ implies } f(x_1, \ldots, x_n) = f(y_1, \ldots, y_n)$$

$$a = b,\ v_1 = v_2 \text{ implies } f(a, v_1) = f(b, v_2)$$

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:

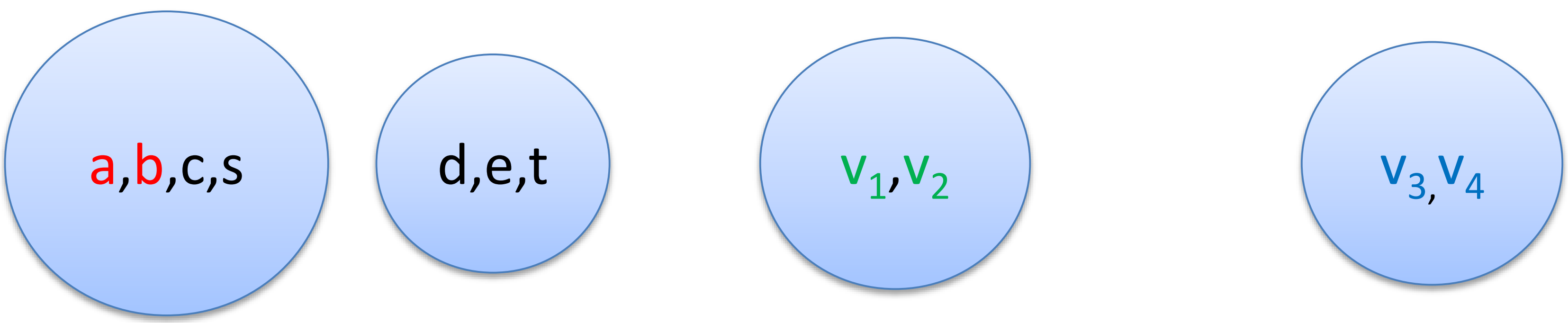
$x_1 = y_1, \ldots, x_n = y_n$ implies $f(x_1, \ldots, x_n) = f(y_1, \ldots, y_n)$

$a = b,\ v_1 = v_2$ 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:

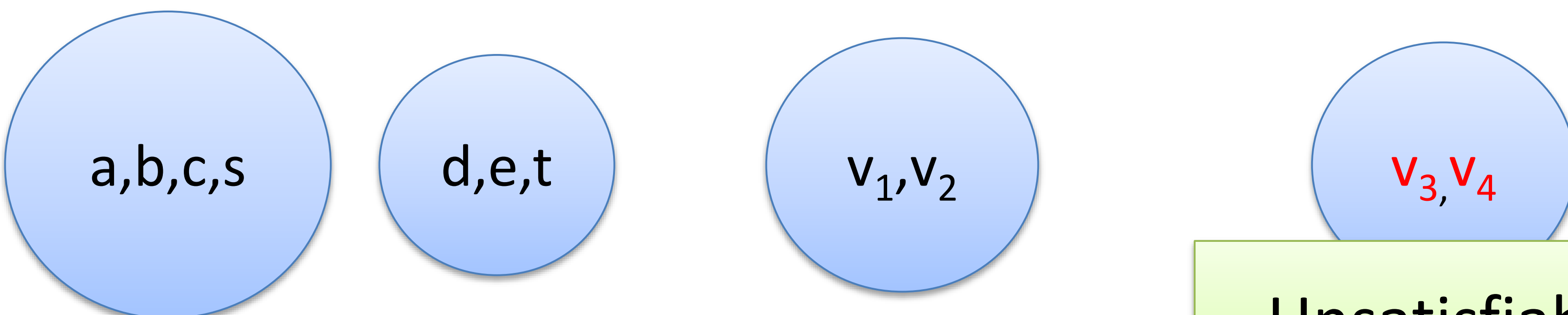
$x_1 = y_1, \ldots, x_n = y_n$ implies $f(x_1, \ldots, x_n) = f(y_1, \ldots, y_n)$

$a = b,\ v_1 = v_2$ implies $v_3 = v_4$

Deciding Equality + (uninterpreted) Functions

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

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



Unsatisfiable

Congruence Rule:

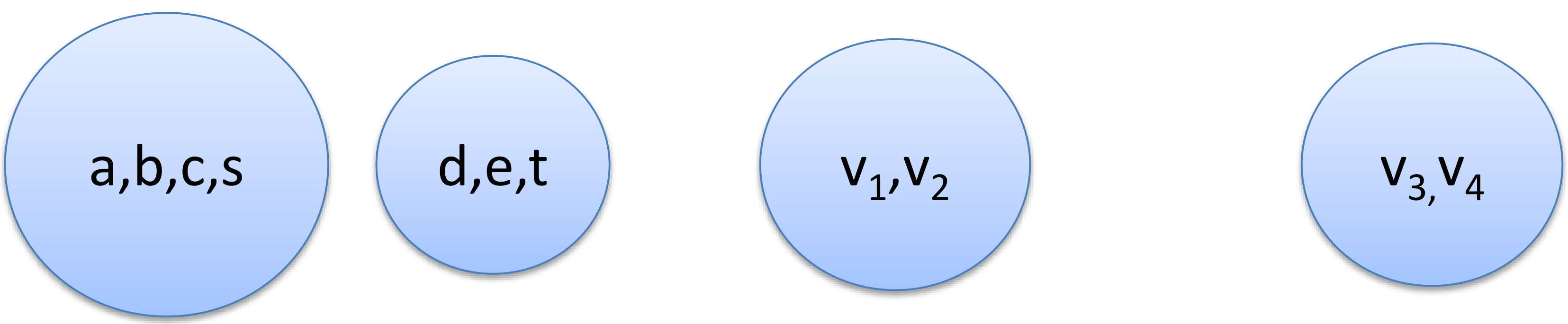
$x_1 = y_1, \ldots, x_n = y_n$ implies $f(x_1, \ldots, x_n) = f(y_1, \ldots, y_n)$

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



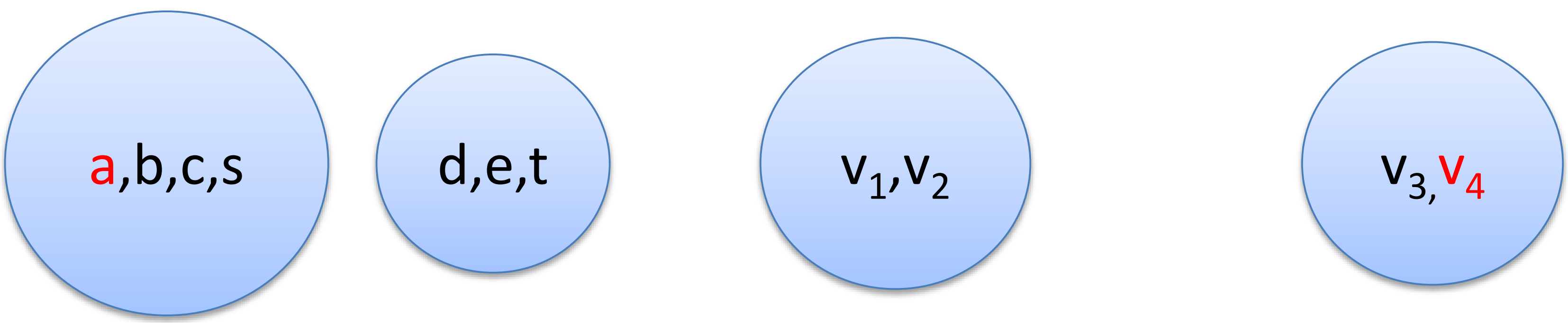
Congruence Rule:

$x_1 = y_1, \ldots, x_n = y_n$ implies $f(x_1, \ldots, x_n) = f(y_1, \ldots, y_n)$

Deciding Equality + (uninterpreted) Functions

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

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



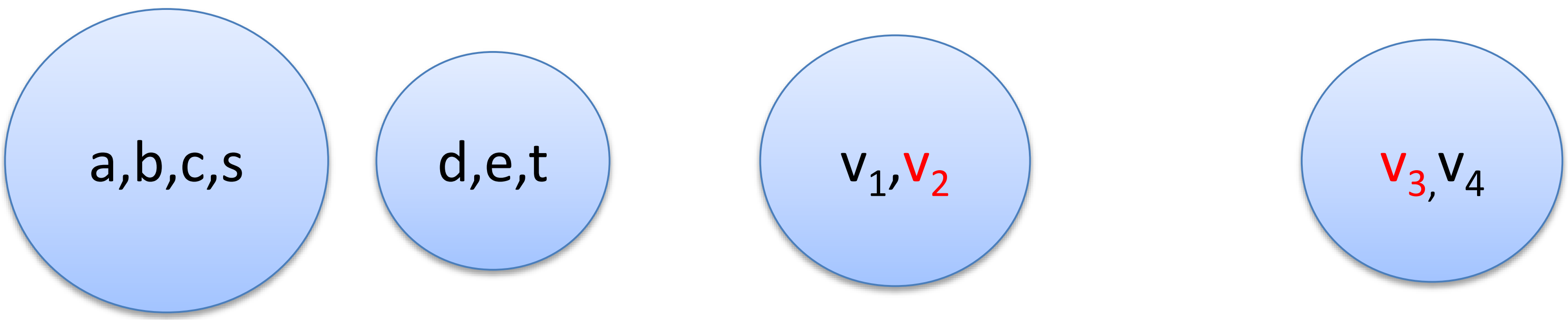
Congruence Rule:

$x_1 = y_1, \ldots, x_n = y_n$ implies $f(x_1, \ldots, x_n) = f(y_1, \ldots, y_n)$

Deciding Equality + (uninterpreted) Functions

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

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



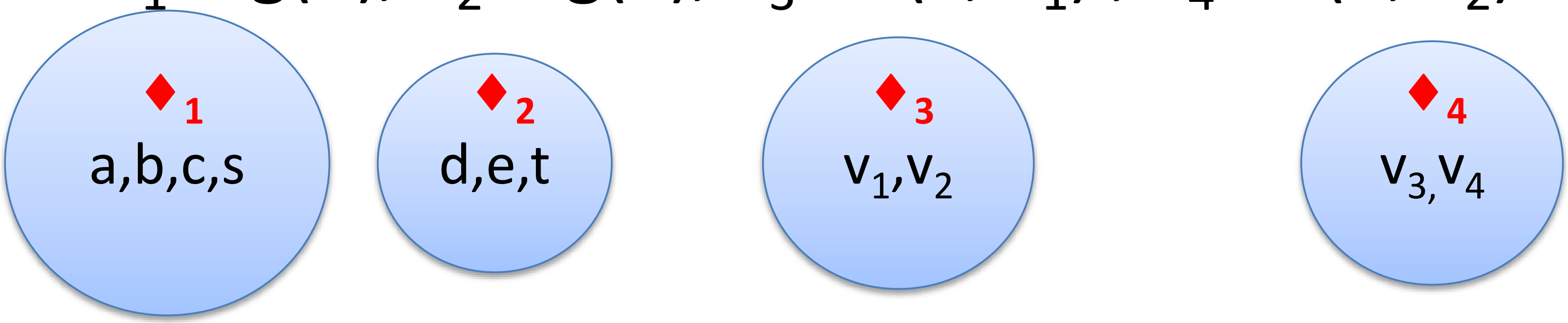
Congruence Rule:

$x_1 = y_1, \ldots, x_n = y_n$ 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:

$|M| = \{\blacklozenge_1, \blacklozenge_2, \blacklozenge_3, \blacklozenge_4\}$

$M(a) = M(b) = M(c) = M(s) = \blacklozenge_1$

$M(d) = M(e) = M(t) = \blacklozenge_2$

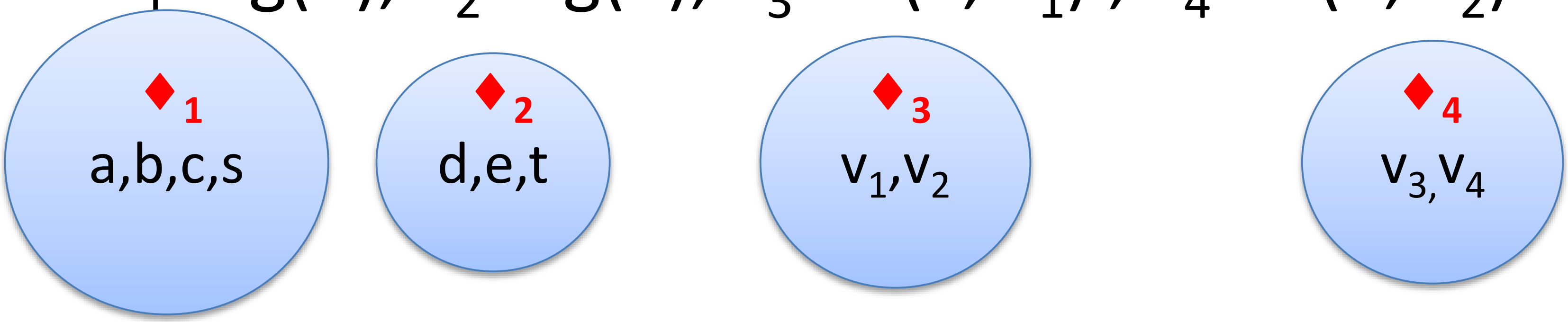
$M(v_1) = M(v_2) = \blacklozenge_3$

$M(v_3) = M(v_4) = \blacklozenge_4$

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:

$$|M| = \{\blacklozenge_1, \blacklozenge_2, \blacklozenge_3, \blacklozenge_4\}$$

$$M(a) = M(b) = M(c) = M(s) = \blacklozenge_1$$

$$M(d) = M(e) = M(t) = \blacklozenge_2$$

$$M(v_1) = M(v_2) = \blacklozenge_3$$

$$M(v_3) = M(v_4) = \blacklozenge_4$$

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:

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

$$M(a) = \blacklozenge_i$$

$$M(v) = \blacklozenge_j$$

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

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:

$$M(g)(\blacklozenge_i) = \blacklozenge_j \text{ if there is } v \equiv g(a) \text{ s.t. } M(a) = \blacklozenge_i,\ M(v) = \blacklozenge_j$$

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

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

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

$M(a) = M(b) = \blacklozenge_1$ and $M(v) = \blacklozenge_2 \neq \blacklozenge_3 = M(w)$

So, is $M(g)(\blacklozenge_1) = \blacklozenge_2$ or $M(g)(\blacklozenge_1) = \blacklozenge_3$?

Deciding Equality + (uninterpreted) Functions

Building the interpretation for function symbols

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

$M(g)(\blacklozenge_i) = \blacklozenge_j$ if there is $v \equiv g(a)$

$M(a) = \blacklozenge_i$

$M(v) = \blacklozenge_j$

$= \blacklozenge_k$, otherwise ($\blacklozenge_k$ is an arbitrary element)

This is impossible because of the congruence rule!

a and b are in the same "ball", then so are v and w

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

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

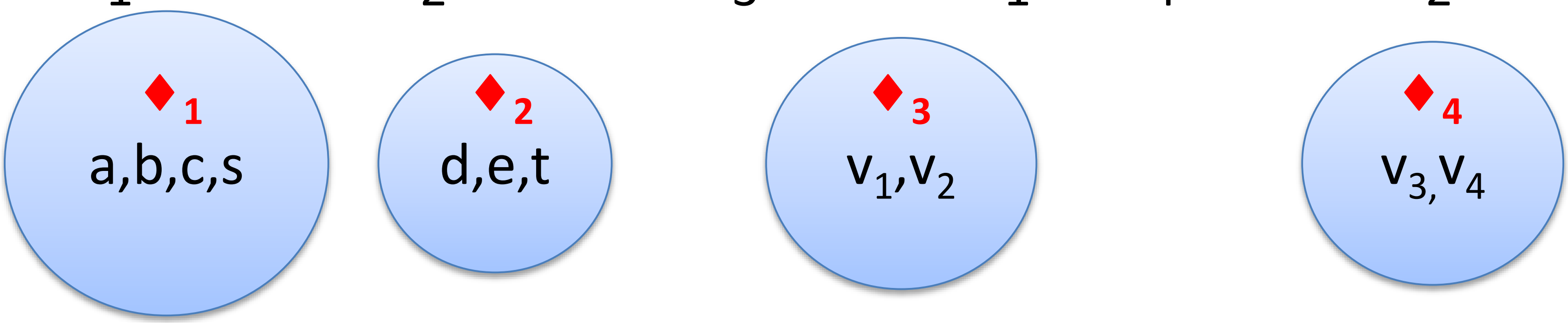
$M(a) = M(b) = \blacklozenge_1$ and $M(v) = \blacklozenge_2 \neq \blacklozenge_3 = M(w)$

So, is $M(g)(\blacklozenge_1) = \blacklozenge_2$ or $M(g)(\blacklozenge_1) = \blacklozenge_3$?

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:

$$|M| = \{\blacklozenge_1, \blacklozenge_2, \blacklozenge_3, \blacklozenge_4\}$$

$$M(a) = M(b) = M(c) = M(s) = \blacklozenge_1$$

$$M(d) = M(e) = M(t) = \blacklozenge_2$$

$$M(v_1) = M(v_2) = \blacklozenge_3$$

$$M(v_3) = M(v_4) = \blacklozenge_4$$

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:

$|M| = \{\blacklozenge_1, \blacklozenge_2, \blacklozenge_3, \blacklozenge_4\}$

$M(a) = M(b) = M(c) = M(s) = \blacklozenge_1$

$M(d) = M(e) = M(t) = \blacklozenge_2$

$M(v_1) = M(v_2) = \blacklozenge_3$

$M(v_3) = M(v_4) = \blacklozenge_4$

$M(g)(\blacklozenge_i) = \blacklozenge_j$ if there is $v \equiv g(a)$ s.t.

$M(a) = \blacklozenge_i$

$M(v) = \blacklozenge_j$

$= \blacklozenge_k$, 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:

$$|M| = \{\blacklozenge_1, \blacklozenge_2, \blacklozenge_3, \blacklozenge_4\}$$

$$M(a) = M(b) = M(c) = M(s) = \blacklozenge_1$$

$$M(d) = M(e) = M(t) = \blacklozenge_2$$

$$M(v_1) = M(v_2) = \blacklozenge_3$$

$$M(v_3) = M(v_4) = \blacklozenge_4$$

$$M(g) = \{\blacklozenge_2 \rightarrow \blacklozenge_3\}$$

$M(g)(\blacklozenge_i) = \blacklozenge_j$ if there is $v \equiv g(a)$ s.t.

$$M(a) = \blacklozenge_i$$

$$M(v) = \blacklozenge_j$$

$= \blacklozenge_k$, 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:

$|M| = \{\blacklozenge_1, \blacklozenge_2, \blacklozenge_3, \blacklozenge_4\}$

$M(a) = M(b) = M(c) = M(s) = \blacklozenge_1$

$M(d) = M(e) = M(t) = \blacklozenge_2$

$M(v_1) = M(v_2) = \blacklozenge_3$

$M(v_3) = M(v_4) = \blacklozenge_4$

$M(g) = \{\blacklozenge_2 \rightarrow \blacklozenge_3\}$

$M(g)(\blacklozenge_i) = \blacklozenge_j$ if there is $v \equiv g(a)$ s.t.

$M(a) = \blacklozenge_i$

$M(v) = \blacklozenge_j$

$= \blacklozenge_k$, 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:

$|M| = \{\blacklozenge_1, \blacklozenge_2, \blacklozenge_3, \blacklozenge_4\}$

$M(a) = M(b) = M(c) = M(s) = \blacklozenge_1$

$M(d) = M(e) = M(t) = \blacklozenge_2$

$M(v_1) = M(v_2) = \blacklozenge_3$

$M(v_3) = M(v_4) = \blacklozenge_4$

$M(g) = \{\blacklozenge_2 \rightarrow \blacklozenge_3, \text{ else} \rightarrow \blacklozenge_1\}$

$M(g)(\blacklozenge_i) = \blacklozenge_j$ if there is $v \equiv g(a)$ s.t.

$M(a) = \blacklozenge_i$

$M(v) = \blacklozenge_j$

$= \blacklozenge_k$, 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:

$|M| = \{\blacklozenge_1, \blacklozenge_2, \blacklozenge_3, \blacklozenge_4\}$

$M(a) = M(b) = M(c) = M(s) = \blacklozenge_1$

$M(d) = M(e) = M(t) = \blacklozenge_2$

$M(v_1) = M(v_2) = \blacklozenge_3$

$M(v_3) = M(v_4) = \blacklozenge_4$

$M(g) = \{\blacklozenge_2 \rightarrow \blacklozenge_3,\ \text{else} \rightarrow \blacklozenge_1\}$

$M(f) = \{ (\blacklozenge_1, \blacklozenge_3) \rightarrow \blacklozenge_4,\ \text{else} \rightarrow \blacklozenge_1\}$

$M(g)(\blacklozenge_i) = \blacklozenge_j$ if there is $v \equiv g(a)$ s.t.

$M(a) = \blacklozenge_i$

$M(v) = \blacklozenge_j$

$= \blacklozenge_k$, otherwise

Deciding Equality + (uninterpreted) Functions

What about predicates?

$$p(a, b), \quad \neg p(c, b)$$

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:

case-analysis

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

Case Analysis

Many verification/analysis problems require:

case-analysis

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

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:

case-analysis

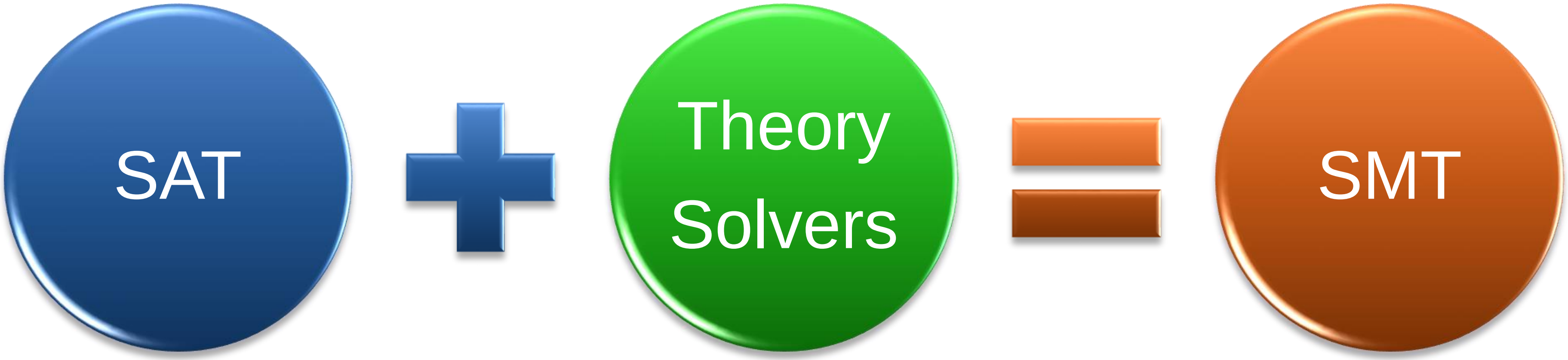
$x \geq 0, y = x + 1, (y > 2 \vee y < 1)$

Naïve Solution: Convert to DNF

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

Too Inefficient!
(exponential blowup)

SMT : Basic Architecture



Case Analysis

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

DPLL

M | F

Partial model

Set of clauses

DPLL

Guessing

$$p \mid p \vee q, \neg q \vee r$$

$$\Downarrow$$

$$p, \neg q \mid p \vee q, \neg q \vee r$$

DPLL

Deducing

$p \mid p \vee q, \neg p \vee s$

$\Downarrow$

$p, s \mid p \vee q, \neg p \vee s$

DPLL

Backtracking

$$p, \neg s, q \mid p \vee q, s \vee q, \neg p \vee \neg q$$

$$\Downarrow$$

$$p, s \mid p \vee q, s \vee q, \neg p \vee \neg q$$

SAT + Theory solvers

Basic Idea

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

Abstract (aka "naming" atoms)

$p_1,\ p_2,\ (p_3 \vee p_4)$

$p_1 \equiv (x \geq 0),\ p_2 \equiv (y = x + 1),$

$p_3 \equiv (y > 2),\ p_4 \equiv (y < 1)$

SAT + Theory solvers

Basic Idea

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

Abstract (aka "naming" atoms)

$p_1,\ \ p_2,\ (p_3 \vee p_4)$

$p_1 \equiv (x \geq 0),\ p_2 \equiv (y = x + 1),$

$p_3 \equiv (y > 2),\ p_4 \equiv (y < 1)$

SAT
Solver

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

$p_1,\ p_2,\ \neg p_3,\ p_4$

SAT + Theory solvers

Basic Idea

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

Abstract (aka "naming" atoms)

$p_1,\ p_2,\ (p_3 \vee p_4)$

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

SAT Solver

Assignment
$p_1,\ p_2,\ \neg p_3,\ p_4$

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

SAT + Theory solvers

Basic Idea

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

Abstract (aka “naming” atoms)

$p_1,\ p_2,\ (p_3 \vee p_4)$

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

SAT Solver

Assignment
$p_1,\ p_2,\ \neg p_3,\ p_4$

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

Theory Solver

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

SAT + Theory solvers

Basic Idea

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

Abstract (aka "naming" atoms)

$p_1,\ p_2,\ (p_3 \vee p_4)$

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

SAT Solver

Assignment
$p_1,\ p_2,\ \neg p_3,\ p_4$

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

Theory Solver

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

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

SAT + Theory solvers

New Lemma

$\neg p_1 \vee \neg p_2 \vee \neg p_4$

Unsatisfiable

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

Theory Solver

AKA

Theory conflict

SAT + Theory solvers: Main loop

procedure SmtSolver(F)
    (F_p, M) := Abstract(F)
    loop
        (R, A) := SAT_solver(F_p)
        if R = UNSAT **then return** UNSAT
        S := Concretize(A, M)
        (R, S') := Theory_solver(S)
        if R = SAT **then return** SAT
        L := New_Lemma(S', M)
        Add L to F_p

SAT + Theory solvers

Basic Idea

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

Abstract (aka "naming" atoms)

F$_p$: $p_1,\ p_2,\ (p_3 \vee p_4)$

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

SAT Solver

A: Assignment
$p_1,\ p_2,\ \neg p_3,\ p_4$

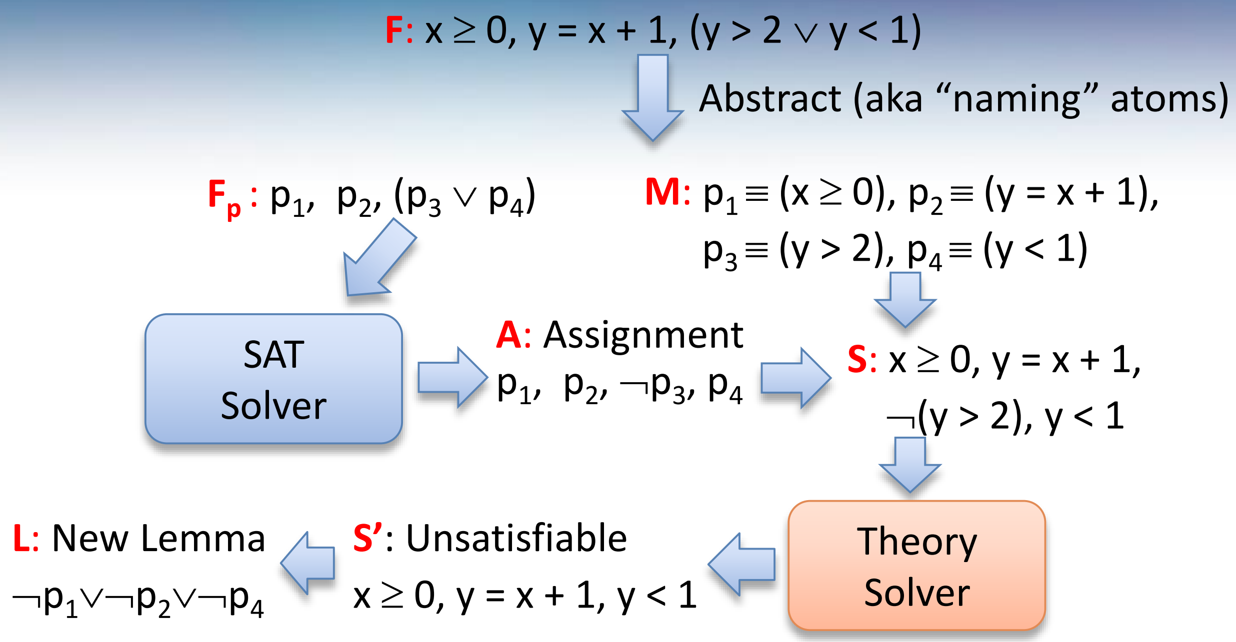
S: $x \geq 0,\ y = x + 1,$
$\neg(y > 2),\ y < 1$

Theory Solver

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

L: New Lemma
$\neg p_1 \vee \neg p_2 \vee \neg p_4$

SAT + Theory solvers



procedure SMT_Solver(F)

(F$_p$, M) := Abstract(F)

loop

(R, A) := SAT_solver(F$_p$)

if R = UNSAT **then return** UNSAT

S = Concretize(A, M)

(R, S') := Theory_solver(S)

if R = SAT **then return** SAT

L := New_Lemma(S, M)

Add L to F$_p$

"Lazy translation"
to
DNF

SAT + Theory solvers

State-of-the-art SMT solvers implement many improvements.

SAT + Theory solvers

Incrementality

Send the literals to the Theory solver as they are assigned by the SAT solver

$p_1 \equiv (x \geq 0)$, $p_2 \equiv (y = x + 1)$,

$p_3 \equiv (y > 2)$, $p_4 \equiv (y < 1)$, $p_5 \equiv (x < 2)$,

$p_1,\ p_2,\ p_4 \ \mid\ p_1,\ p_2, (p_3 \vee p_4), (p_5 \vee \neg p_4)$

Partial assignment is already Theory inconsistent.

SAT + Theory solvers

Efficient Backtracking

We don't want to restart from scratch after each backtracking operation.

SAT + Theory solvers

Efficient Lemma Generation (computing a small S')

Avoid lemmas containing redundant literals.

$p_1 \equiv (x \geq 0)$, $p_2 \equiv (y = x + 1)$,

$p_3 \equiv (y > 2)$, $p_4 \equiv (y < 1)$, $p_5 \equiv (x < 2)$,

$p_1,\ p_2,\ p_3,\ p_4 \ \mid\ p_1,\ p_2,\ (p_3 \vee p_4),\ (p_5 \vee \neg p_4)$

$\neg p_1 \vee \neg p_2 \vee \neg p_3 \vee \neg p_4$

Imprecise Lemma

SAT + Theory solvers

Theory Propagation

It is the SMT equivalent of unit propagation.

$p_1 \equiv (x \geq 0)$, $p_2 \equiv (y = x + 1)$,

$p_3 \equiv (y > 2)$, $p_4 \equiv (y < 1)$, $p_5 \equiv (x < 2)$,

$p_1, \; p_2 \mid \; p_1, \; p_2, (p_3 \vee p_4), (p_5 \vee \neg p_4)$

$\Downarrow$ p_1, p_2 imply $\neg p_4$ by theory propagation

$p_1, \; p_2, \neg p_4 \mid \; p_1, \; p_2, (p_3 \vee p_4), (p_5 \vee \neg p_4)$

SAT + Theory solvers

Theory Propagation

It is the SMT equivalent of unit propagation.

$p_1 \equiv (x \geq 0)$, $p_2 \equiv (y = x + 1)$,

$p_3 \equiv (y > 2)$, $p_4 \equiv (y < 1)$, $p_5 \equiv (x < 2)$,

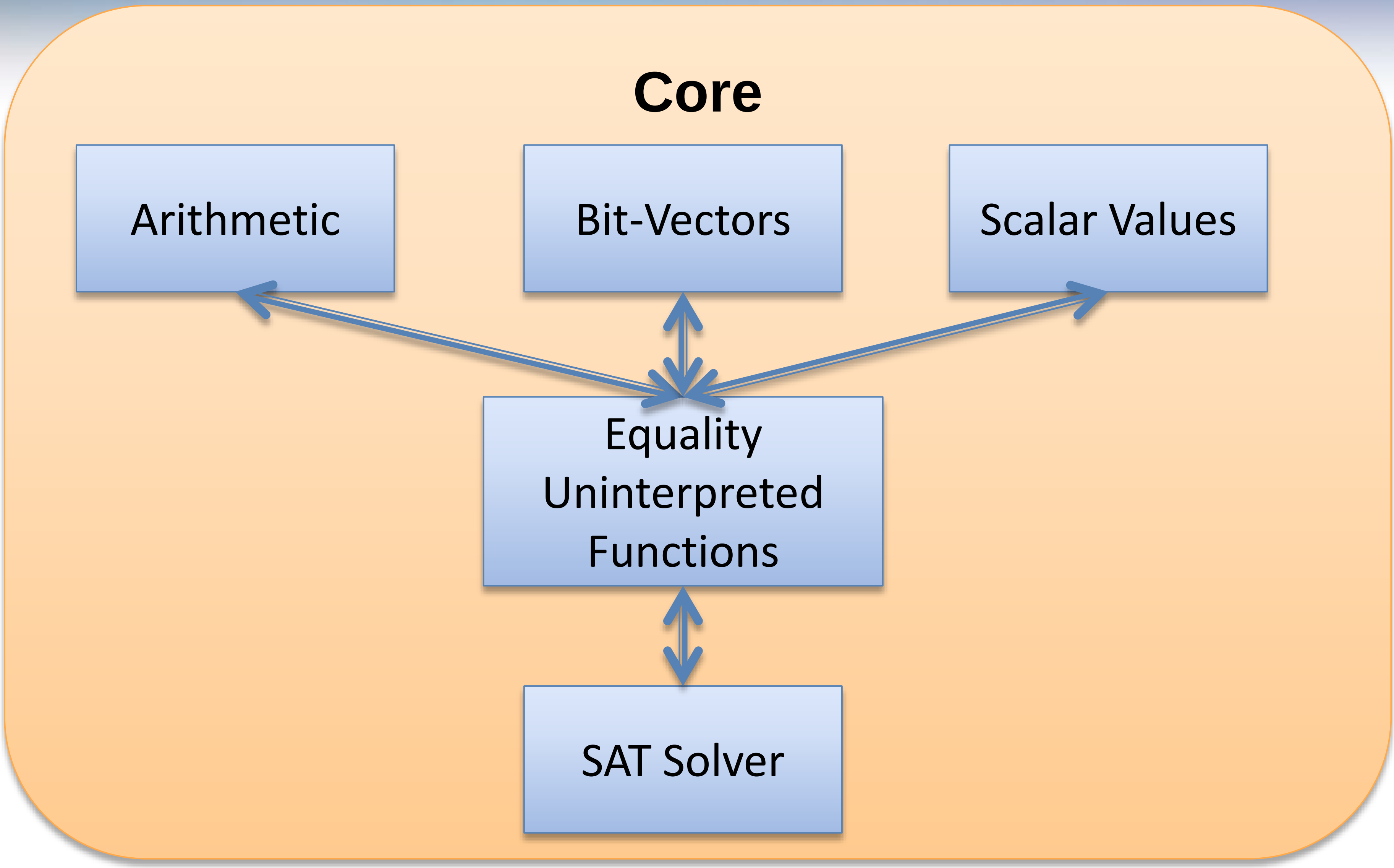
$p_1,\ p_2 \mid\ p_1,\ p_2, (p_3 \vee p_4), (p_5 \vee \neg p_4)$

$\Downarrow$ p_1, p_2 imply $\neg p_4$ by theory propagation

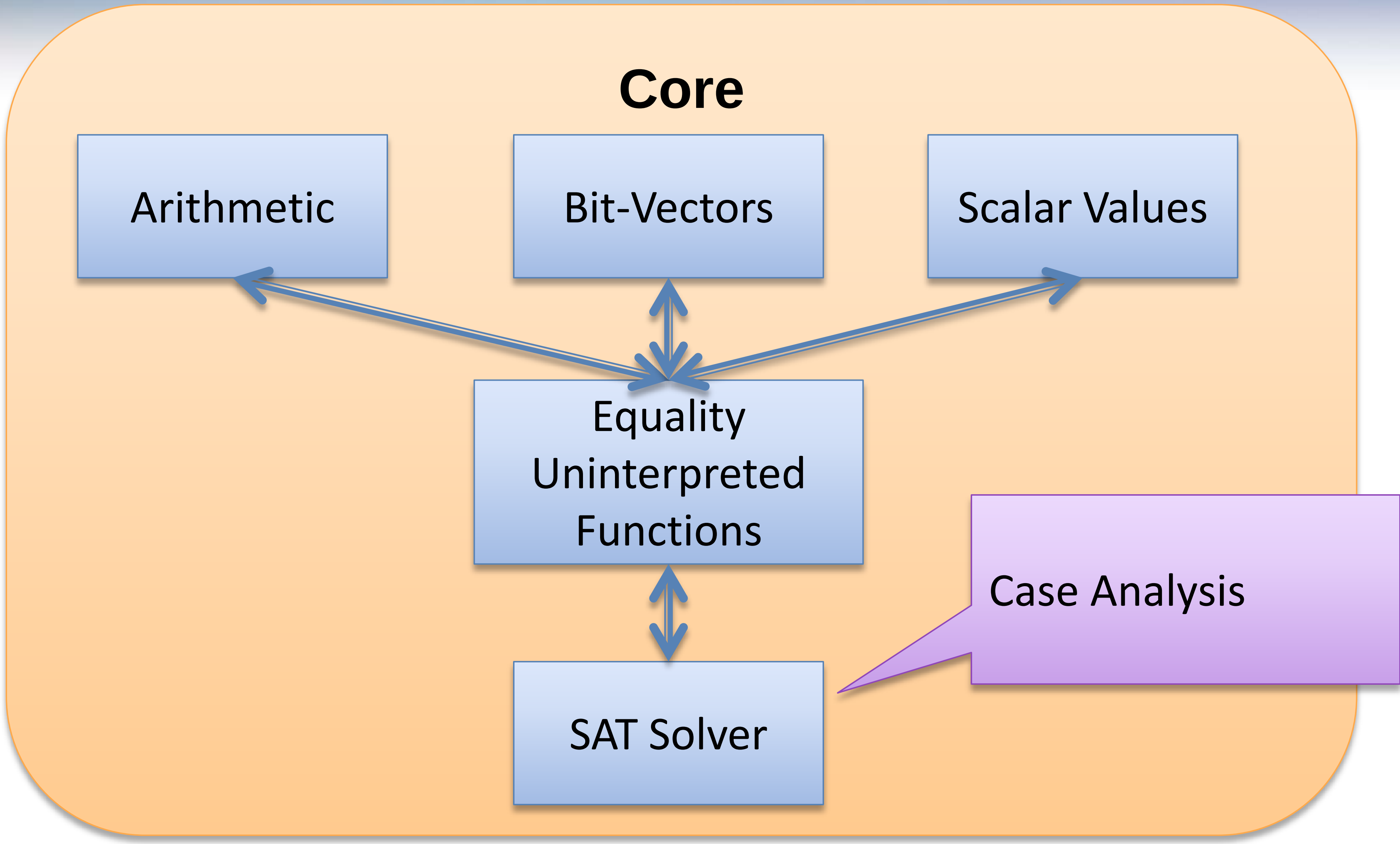
$p_1,\ p_2,\ \neg p_4 \mid\ p_1,\ p_2, (p_3 \vee p_4), (p_5 \vee \neg p_4)$

Tradeoff between precision × performance.

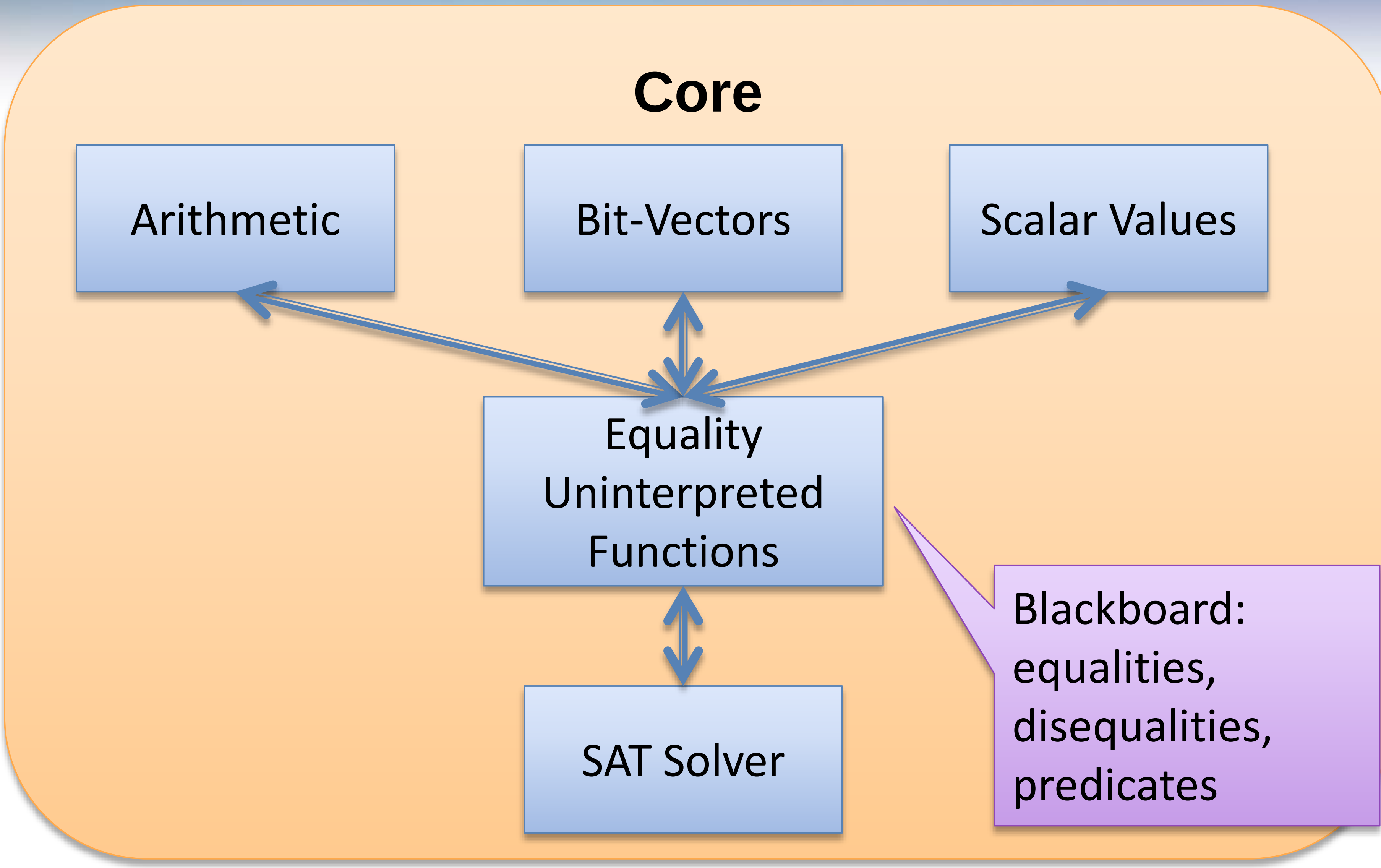
An Architecture: the core



An Architecture: the core



An Architecture: the core



Combining Theories

In practice, we need a combination of theories.

$b + 2 = c$ and $f(read(write(a,b,3), c-2)) \neq f(c-b+1)$

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

Main questions:

Is the union of two theories $T1 \cup T2$ consistent?

Given a solvers for T1 and T2, how can we build a solver for $T1 \cup T2$?

Microsoft Research

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.

Disjoint Theories

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: $\{0, -1, 1, -2, 2, \ldots, +, -, *, >, <, \geq, \leq\}$

Array symbols: { read, write }

Purification

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

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

$\Downarrow$

$b + 2 = c,\ v_6 \neq v_7$

$v_1 \equiv 3,\ v_2 \equiv write(a,\ b,\ v_1),\ v_3 \equiv c\text{-}2,\ v_4 \equiv read(v_2,\ v_3),$

$v_5 \equiv c\text{-}b\text{+}1,\ v_6 \equiv f(v_4),\ v_7 \equiv f(v_5)$

Purification

It is a different name for our "naming" subterms procedure.

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

$b + 2 = c,\ v_6 \neq v_7$

$v_1 \equiv 3,\ v_2 \equiv write(a,\ b,\ v_1),\ v_3 \equiv c\text{-}2,\ v_4 \equiv read(v_2,\ v_3),$

$v_5 \equiv c\text{-}b\text{+}1,\ v_6 \equiv f(v_4),\ v_7 \equiv f(v_5)$

$b + 2 = c,\ v_1 \equiv 3,\ v_3 \equiv c\text{-}2,\ v_5 \equiv c\text{-}b\text{+}1,$

$v_2 \equiv write(a,\ b,\ v_1),\ v_4 \equiv read(v_2,\ v_3),$

$v_6 \equiv f(v_4),\ v_7 \equiv f(v_5),\ v_6 \neq v_7$

Stably Infinite Theories

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.

Important Result

The union of two consistent, disjoint, stably infinite theories is consistent.

Convexity

A theory T is convex iff

for all finite sets S of literals and

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

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

iff

S implies $a_i = b_i$ for some $1 \leq i \leq n$

Microsoft Research

Convexity: Results

Every convex theory with non trivial models is stably infinite.

All Horn equational theories are convex.

formulas of the form $s_1 \neq r_1 \vee \ldots \vee s_n \neq r_n \vee t = t'$

Linear rational arithmetic is convex.

Convexity: Negative Results

Linear integer arithmetic is not convex

$1 \leq a \leq 2$, $b = 1$, $c = 2$ implies $a = b \vee a = c$

Nonlinear arithmetic

$a^2 = 1$, $b = 1$, $c = -1$ implies $a = b \vee a = c$

Theory of bit-vectors

Theory of arrays

$c_1 = read(write(a, i, c_2), j)$, $c_3 = read(a, j)$

implies $c_1 = c_2 \vee c_1 = c_3$

Combination of non-convex theories

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 $p_1 \vee \neg p_2 \vee p_3$ add

$$f(a_1, a_2, a_3) \neq f(0, 1, 0)$$

Microsoft Research

Combination of non-convex theories

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 $p_1 \vee \neg p_2 \vee p_3$ add

$$f(a_1, a_2, a_3) \neq f(0, 1, 0)$$

implies

$$a_1 \neq 0 \vee a_2 \neq 1 \vee a_3 \neq 0$$

Microsoft Research

Nelson-Oppen Combination

Let $\mathcal{T}_1$ and $\mathcal{T}_2$ 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 $\mathcal{T}$ is consistent and stably infinite.
2. Satisfiability of quantifier free conjunction of literals in $\mathcal{T}$ can be decided in $O(2^{n^2} \times (T_1(n) + T_2(n))$.
3. If $\mathcal{T}_1$ and $\mathcal{T}_2$ are convex, then so is $\mathcal{T}$ and satisfiability in $\mathcal{T}$ is in $O(n^3 \times (T_1(n) + T_2(n)))$.

Microsoft Research

Nelson-Oppen Combination

The combination procedure:

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

Purification: Preserving satisfiability transform ϕ into $\phi_1 \wedge \phi_2$, 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 ψ.

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 $\phi_i \wedge \psi$ is satisfiable.

Return: If both return yes, return yes. No, otherwise.

Microsoft Research

NO deterministic procedure (for convex theories)

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

Component Procedures : Use individual procedures to decide whether ϕ_i is satisfiable.

Remark: $\mathcal{T}_i \vdash (\phi_i \Rightarrow x = y)$ iff $\phi_i \wedge x \neq y$ is not satisfiable in $\mathcal{T}_i$.

NO deterministic procedure Completeness

Assume the theories are convex.

- Suppose ϕ_i is satisfiable.
- Let E be the set of equalities $x_j = x_k$ $(j \neq k)$ such that, $\mathcal{T}_i \not\vdash \phi_i \Rightarrow x_j = x_k$.
- By convexity, $\mathcal{T}_i \not\vdash \phi_i \Rightarrow \bigvee_E x_j = x_k$.
- $\phi_i \wedge \bigwedge_E x_j \neq x_k$ 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.

Microsoft Research

NO procedure: Example

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

Arithmetic

$b + 2 = c,$

$v_1 \equiv 3,$

$v_3 \equiv c\text{-}2,$

$v_5 \equiv c\text{-}b\text{+}1$

Arrays

$v_2 \equiv write(a,\ b,\ v_1),$

$v_4 \equiv read(v_2,\ v_3)$

EUF

$v_6 \equiv f(v_4),$

$v_7 \equiv f(v_5),$

$v_6 \neq v_7$

NO procedure: Example

$b + 2 = c, f(read(write(a,b,3), c-2)) \neq f(c-b+1)$

Arithmetic

$b + 2 = \mathbf{c},$

$v_1 \equiv 3,$

$v_3 \equiv c-2,$

$v_5 \equiv c-b+1$

Arrays

$v_2 \equiv write(a, b, v_1),$

$v_4 \equiv read(v_2, v_3)$

EUF

$v_6 \equiv f(v_4),$

$v_7 \equiv f(v_5),$

$v_6 \neq v_7$

Substituting c

NO procedure: Example

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

Arithmetic

$b + 2 = c,$

$v_1 \equiv 3,$

$\mathbf{v_3 \equiv b},$

$v_5 \equiv 3$

Arrays

$v_2 \equiv write(a, b, v_1),$

$v_4 \equiv read(v_2, v_3),$

EUF

$v_6 \equiv f(v_4),$

$v_7 \equiv f(v_5),$

$v_6 \neq v_7$

Propagating $v_3 = b$

NO procedure: Example

$b + 2 = c, f(read(write(a,b,3), c-2)) \neq f(c-b+1)$

Arithmetic

$b + 2 = c,$

$v_1 \equiv 3,$

$v_3 \equiv b,$

$v_5 \equiv 3$

Arrays

$\mathbf{v_2} \equiv write(a, \mathbf{b}, v_1),$

$v_4 \equiv read(\mathbf{v_2}, \mathbf{v_3}),$

$v_3 = b$

EUF

$v_6 \equiv f(v_4),$

$v_7 \equiv f(v_5),$

$v_6 \neq v_7,$

$v_3 = b$

Deducing $v_4 = v_1$

NO procedure: Example

$b + 2 = c,\ f(read(write(a,b,3), c-2)) \neq f(c-b+1)$

Arithmetic	Arrays	EUF
$b + 2 = c,$	$v_2 \equiv write(a, b, v_1),$	$v_6 \equiv f(v_4),$
$v_1 \equiv 3,$	$v_4 \equiv read(v_2, v_3),$	$v_7 \equiv f(v_5),$
$v_3 \equiv b,$	$v_3 = b,$	$v_6 \neq v_7,$
$v_5 \equiv 3$	$\mathbf{v_4 = v_1}$	$v_3 = b$

Propagating $v_4 = v_1$

Microsoft Research

NO procedure: Example

$b + 2 = c, f(read(write(a,b,3), c-2)) \neq f(c-b+1)$

Arithmetic

$b + 2 = c,$

$\mathbf{v_1 \equiv 3},$

$v_3 \equiv b,$

$\mathbf{v_5 \equiv 3},$

$v_4 = v_1$

Arrays

$v_2 \equiv write(a, b, v_1),$

$v_4 \equiv read(v_2, v_3),$

$v_3 = b,$

$v_4 = v_1$

EUF

$v_6 \equiv f(v_4),$

$v_7 \equiv f(v_5),$

$v_6 \neq v_7,$

$v_3 = b,$

$v_4 = v_1$

Propagating $v_5 = v_1$

NO procedure: Example

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

Arithmetic

$b + 2 = c,$

$v_1 \equiv 3,$

$v_3 \equiv b,$

$v_5 \equiv 3,$

$v_4 = v_1$

Arrays

$v_2 \equiv write(a, b, v_1),$

$v_4 \equiv read(v_2, v_3),$

$v_3 = b,$

$v_4 = v_1$

EUF

$v_6 \equiv f(\mathbf{v_4}),$

$v_7 \equiv f(\mathbf{v_5}),$

$v_6 \neq v_7,$

$v_3 = b,$

$\mathbf{v_4 = v_1},$

$\mathbf{v_5 = v_1}$

Congruence: $v_6 = v_7$

NO procedure: Example

$b + 2 = c,\ f(read(write(a,b,3), c-2)) \neq f(c-b+1)$

Arithmetic

$b + 2 = c,$

$v_1 \equiv 3,$

$v_3 \equiv b,$

$v_5 \equiv 3,$

$v_4 = v_1$

Arrays

$v_2 \equiv write(a, b, v_1),$

$v_4 \equiv read(v_2, v_3),$

$v_3 = b,$

$v_4 = v_1$

EUF

$v_6 \equiv f(v_4),$

$v_7 \equiv f(v_5),$

$\mathbf{v_6 \neq v_7},$

$v_3 = b,$

$v_4 = v_1,$

$v_5 = v_1,$

$\mathbf{v_6 = v_7}$

Unsatisfiable

NO deterministic procedure

Deterministic procedure may fail for non-convex theories.

$0 \leq a \leq 1,\ 0 \leq b \leq 1,\ 0 \leq c \leq 1,$

$f(a) \neq f(b),$

$f(a) \neq f(c),$

$f(b) \neq f(c)$

Microsoft Research

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.

Combining Procedures in Practice

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 $\mathcal{T}_i$ and propagate all equalities implied by the candidate model, hedging that other theories will agree.

$$\textbf{if } M_i \models \mathcal{T}_i \cup \Gamma_i \cup \{u = v\} \textbf{ then } \text{propagate } u = v \ .$$

- 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.

Microsoft Research

Example

$$x = f(y-1), f(x) \neq f(y), 0 \leq x \leq 1, 0 \leq y \leq 1$$

Purifying

Example

$$x = f(z), f(x) \neq f(y), 0 \leq x \leq 1, 0 \leq y \leq 1, z = y - 1$$

Example

$\mathcal{T}_E$			$\mathcal{T}_A$	
Literals	*Eq. Classes*	*Model*	*Literals*	*Model*
$x = f(z)$ $f(x) \neq f(y)$	$\{x, f(z)\}$ $\{y\}$ $\{z\}$ $\{f(x)\}$ $\{f(y)\}$	$E(x) = *_1$ $E(y) = *_2$ $E(z) = *_3$ $E(f) = \{*_1 \mapsto *_4,$ $*_2 \mapsto *_5,$ $*_3 \mapsto *_1,$ *else* $\mapsto *_6\}$	$0 \leq x \leq 1$ $0 \leq y \leq 1$ $z = y - 1$	$A(x) = 0$ $A(y) = 0$ $A(z) = -1$

Assume x = y

Example

$\mathcal{T}_E$			$\mathcal{T}_A$	
Literals	*Eq. Classes*	*Model*	*Literals*	*Model*
$x = f(z)$	$\{x, y, f(z)\}$	$E(x) = *_1$	$0 \le x \le 1$	$A(x) = 0$
$f(x) \neq f(y)$	$\{z\}$	$E(y) = *_1$	$0 \le y \le 1$	$A(y) = 0$
$x = y$	$\{f(x), f(y)\}$	$E(z) = *_2$	$z = y - 1$	$A(z) = -1$
		$E(f) = \{*_1 \mapsto *_3,$ $*_2 \mapsto *_1,$ *else* $\mapsto *_4\}$	$x = y$	

Unsatisfiable

Example

$\mathcal{T}_E$			$\mathcal{T}_A$	
Literals	*Eq. Classes*	*Model*	*Literals*	*Model*
$x = f(z)$	$\{x, f(z)\}$	$E(x) = *_1$	$0 \leq x \leq 1$	$A(x) = 0$
$f(x) \neq f(y)$	$\{y\}$	$E(y) = *_2$	$0 \leq y \leq 1$	$A(y) = 0$
$x \neq y$	$\{z\}$	$E(z) = *_3$	$z = y - 1$	$A(z) = -1$
	$\{f(x)\}$	$E(f) = \{*_1 \mapsto *_4,$	$x \neq y$	
	$\{f(y)\}$	$*_2 \mapsto *_5,$		
		$*_3 \mapsto *_1,$		
		else $\mapsto *_6\}$		

Backtrack, and assert $x \neq y$.

$\mathcal{T}_A$ model need to be fixed.

Example

$\mathcal{T}_E$			$\mathcal{T}_A$	
Literals	*Eq. Classes*	*Model*	*Literals*	*Model*
$x = f(z)$	$\{x, f(z)\}$	$E(x) = *_1$	$0 \leq x \leq 1$	$A(x) = 0$
$f(x) \neq f(y)$	$\{y\}$	$E(y) = *_2$	$0 \leq y \leq 1$	$A(y) = 1$
$x \neq y$	$\{z\}$	$E(z) = *_3$	$z = y - 1$	$A(z) = 0$
	$\{f(x)\}$	$E(f) = \{*_1 \mapsto *_4,$	$x \neq y$	
	$\{f(y)\}$	$*_2 \mapsto *_5,$		
		$*_3 \mapsto *_1,$		
		else $\mapsto *_6\}$		

Assume x = z

Example

$\mathcal{T}_E$			$\mathcal{T}_A$	
Literals	*Eq. Classes*	*Model*	*Literals*	*Model*
$x = f(z)$	$\{x, z,$	$E(x) = *_1$	$0 \leq x \leq 1$	$A(x) = 0$
$f(x) \neq f(y)$	$f(x), f(z)\}$	$E(y) = *_2$	$0 \leq y \leq 1$	$A(y) = 1$
$x \neq y$	$\{y\}$	$E(z) = *_1$	$z = y - 1$	$A(z) = 0$
$x = z$	$\{f(y)\}$	$E(f) = \{*_1 \mapsto *_1,$	$x \neq y$	
		$*_2 \mapsto *_3,$	$x = z$	
		else $\mapsto *_4\}$		

Satisfiable

Example

$\mathcal{T}_E$			$\mathcal{T}_A$	
Literals	*Eq. Classes*	*Model*	*Literals*	*Model*
$x = f(z)$	$\{x, z,$	$E(x) = *_1$	$0 \leq x \leq 1$	$A(x) = 0$
$f(x) \neq f(y)$	$f(x), f(z)\}$	$E(y) = *_2$	$0 \leq y \leq 1$	$A(y) = 1$
$x \neq y$	$\{y\}$	$E(z) = *_1$	$z = y - 1$	$A(z) = 0$
$x = z$	$\{f(y)\}$	$E(f) = \{*_1 \mapsto *_1,$	$x \neq y$	
		$*_2 \mapsto *_3,$	$x = z$	
		else $\mapsto *_4\}$		

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

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

Example

$\mathcal{T}_E$		$\mathcal{T}_A$	
Literals	*Model*	*Literals*	*Model*
$x = f(z)$	$E(x) = *_1$	$0 \leq x \leq 1$	$A(x) = 0$
$f(x) \neq f(y)$	$E(y) = *_2$	$0 \leq y \leq 1$	$A(y) = 1$
$x \neq y$	$E(z) = *_1$	$z = y - 1$	$A(z) = 0$
$x = z$	$E(f) = \{*_1 \mapsto *_1,$	$x \neq y$	$A(f) = \{0 \mapsto 0$
	$*_2 \mapsto *_3,$	$x = z$	$1 \mapsto -1$
	else $\mapsto *_4\}$		*else* $\mapsto 2\}$

Extending A using h.

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