Microsoft Research

Satisfiability Modulo Theories (SMT): ideas and applications

Università Degli Studi Di Milano
Scuola di Dottorato in Informatica, 2010

Leonardo de Moura
Microsoft Research

Satisfiability Modulo Theories (SMT)

Is formula *F* satisfiable modulo theory *T* ?

SMT solvers have specialized algorithms for *T*

Microsoft Research

Satisfiability Modulo Theories (SMT)

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

Satisfiability Modulo Theories (SMT)

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

Arithmetic

Satisfiability Modulo Theories (SMT)

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

Array Theory

Microsoft® Research

Satisfiability Modulo Theories (SMT)

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

Uninterpreted Functions

Satisfiability Modulo Theories (SMT)

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

Substituting c by b+2

Microsoft Research

Satisfiability Modulo Theories (SMT)

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

Simplifying

Satisfiability Modulo Theories (SMT)

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

Satisfiability Modulo Theories (SMT)

b + 2 = c and f(read(write(a,b,3), b)) ≠ f(3)

Applying array theory axiom

forall a,i,v: read(write(a,i,v), i) = v

Satisfiability Modulo Theories (SMT)

$b + 2 = c$ and $f(3) \neq f(3)$

Inconsistent

Microsoft® Research

SMT-Lib

- Repository of Benchmarks
- http://www.smtlib.org
- Benchmarks are divided in "logics":
  - QF_UF: unquantified formulas built over a signature of uninterpreted sort, function and predicate symbols.
  - QF_UFLIA: unquantified linear integer arithmetic with uninterpreted sort, function, and predicate symbols.
  - AUFLIA: closed linear formulas over the theory of integer arrays with free sort, function and predicate symbols.

Ground formulas

For most SMT solvers: ***F is a set of ground formulas***

Many Applications

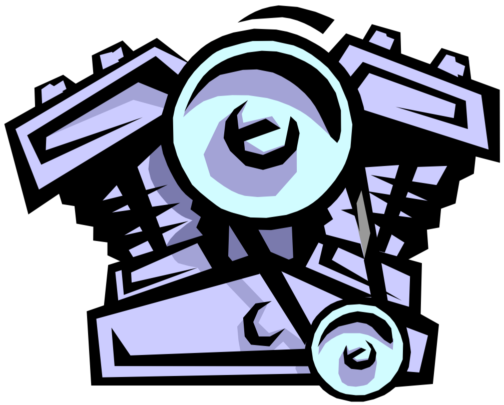
Bounded Model Checking

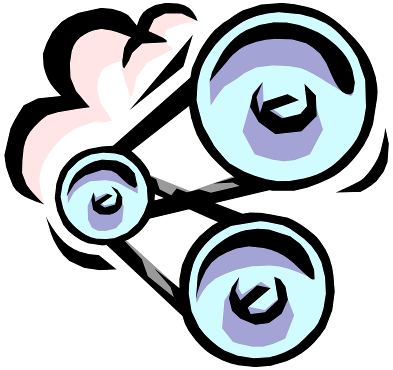
Test-Case Generation

Little Engines of Proof

An SMT Solver is a collection of

Little Engines of Proof



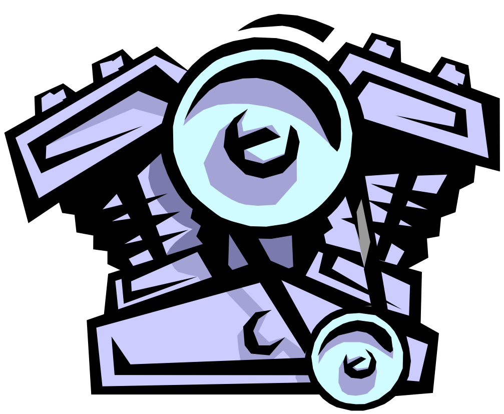


Microsoft Research

Little Engines of Proof

An SMT Solver is a collection of

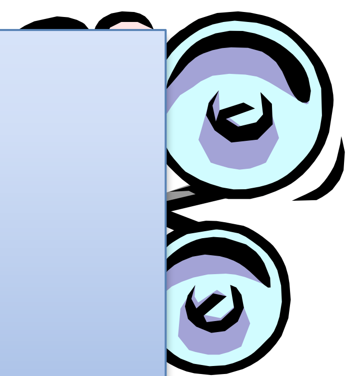
Little Engines of Proof



Examples:

SAT Solver

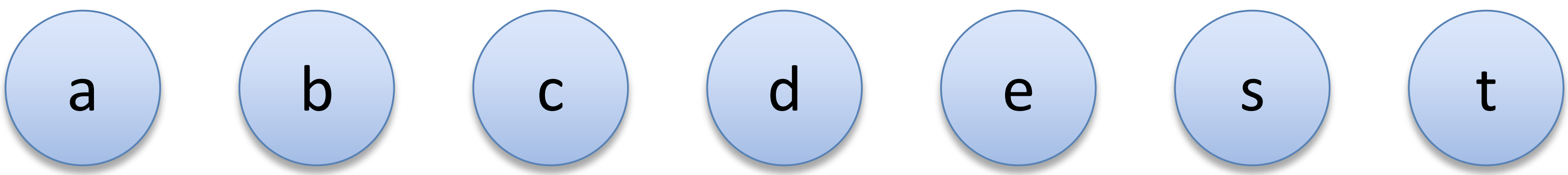
Equality solver



Microsoft Research

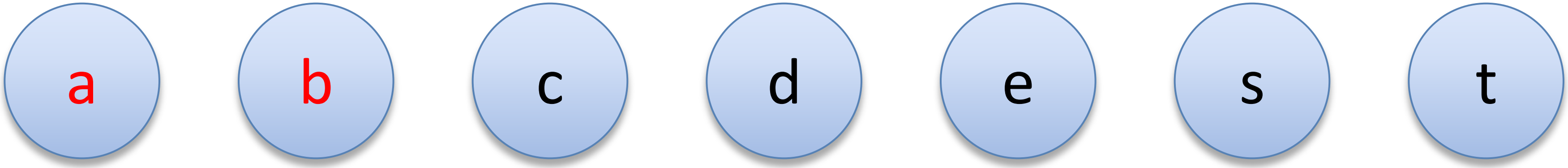
Deciding Equality

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



Deciding Equality

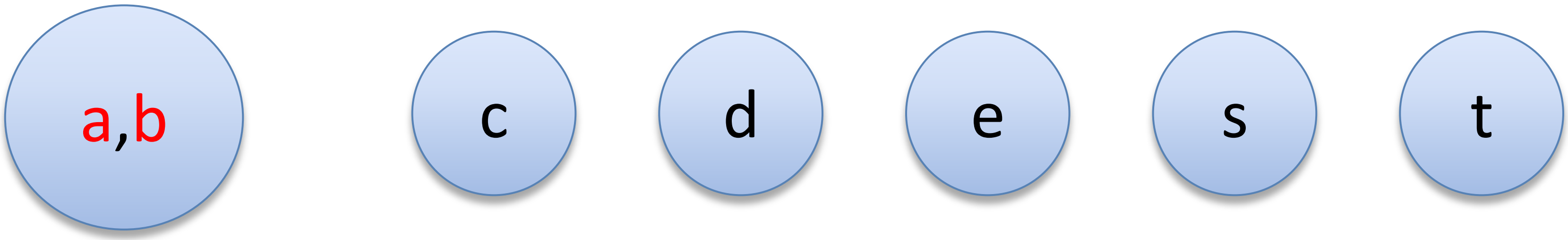
$a = b, b = c, d = e, b = s, d = t, a \neq e, a \neq s$



Microsoft Research

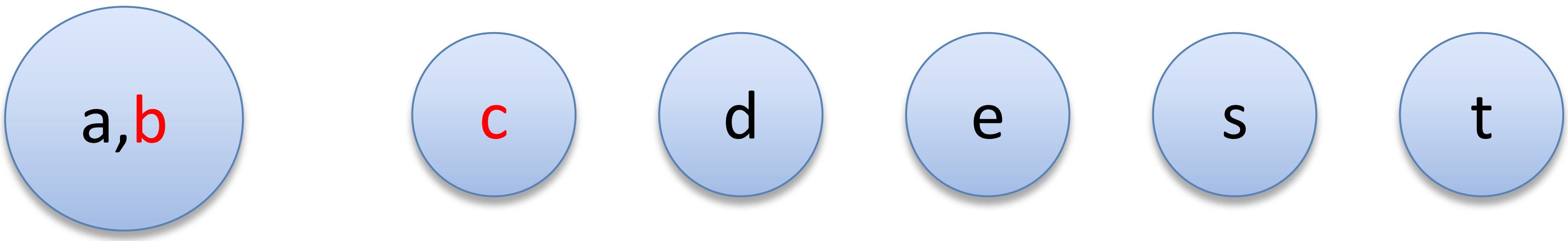
Deciding Equality

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



Deciding Equality

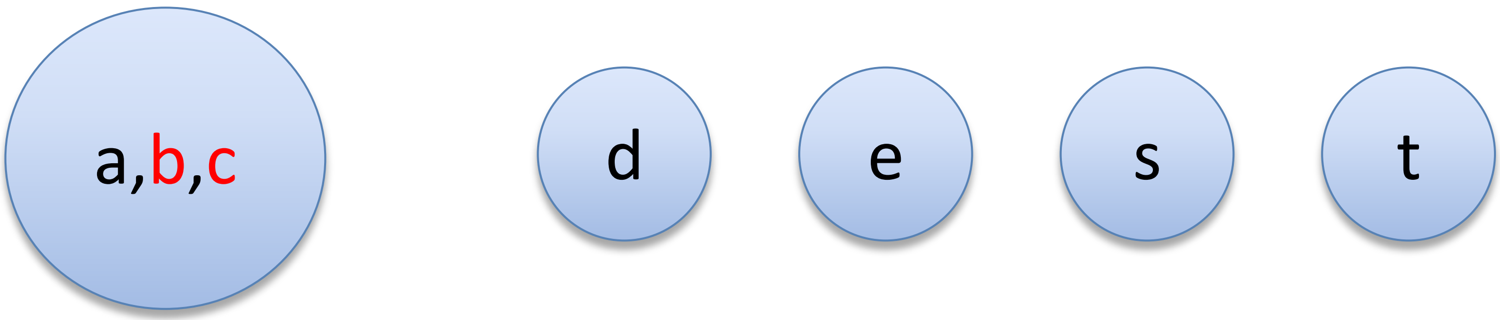
$a = b,\ b = c,\ d = e,\ b = s,\ d = t,\ a \neq e,\ a \neq s$



Microsoft Research

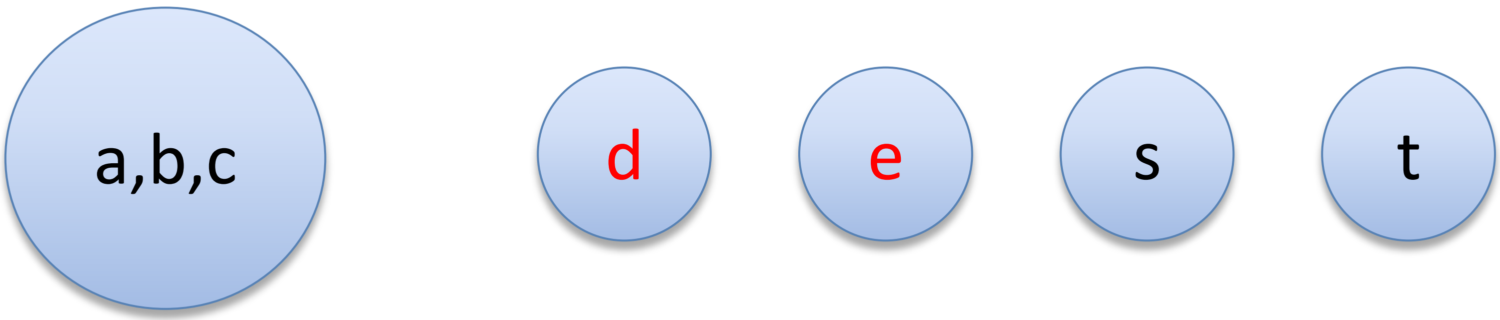
Deciding Equality

a = b, b = c, d = e, b = s, d = t, a ≠ e, a ≠ s



Deciding Equality

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



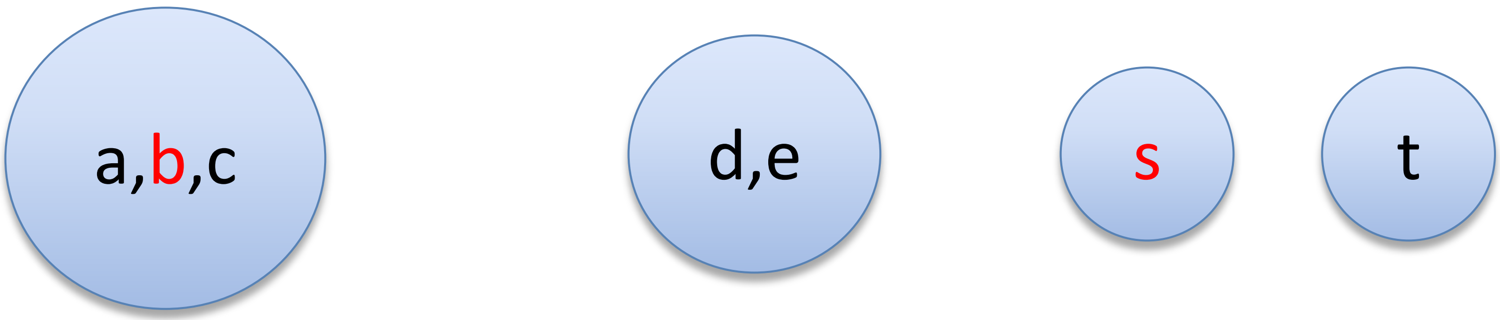
Deciding Equality

a = b, b = c, d = e, b = s, d = t, a ≠ e, a ≠ s

Microsoft Research

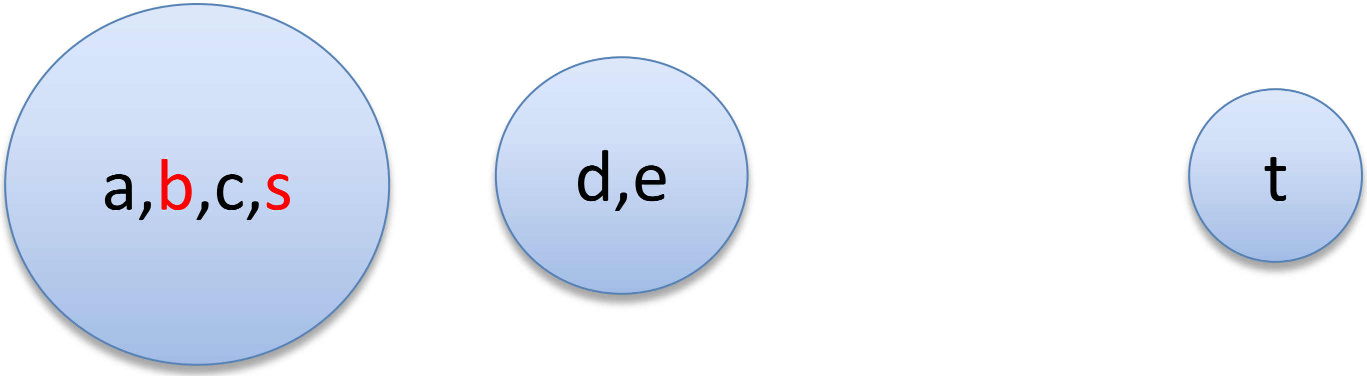
Deciding Equality

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



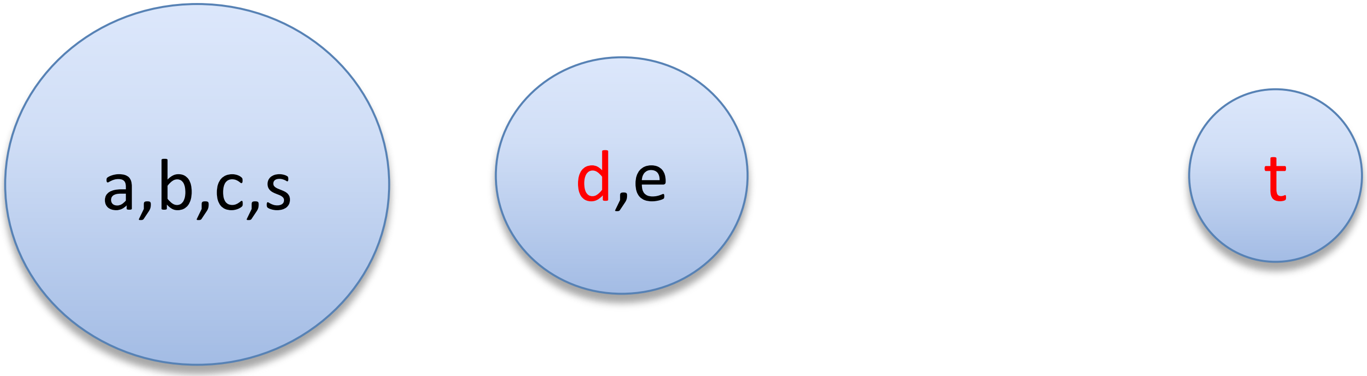
Deciding Equality

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



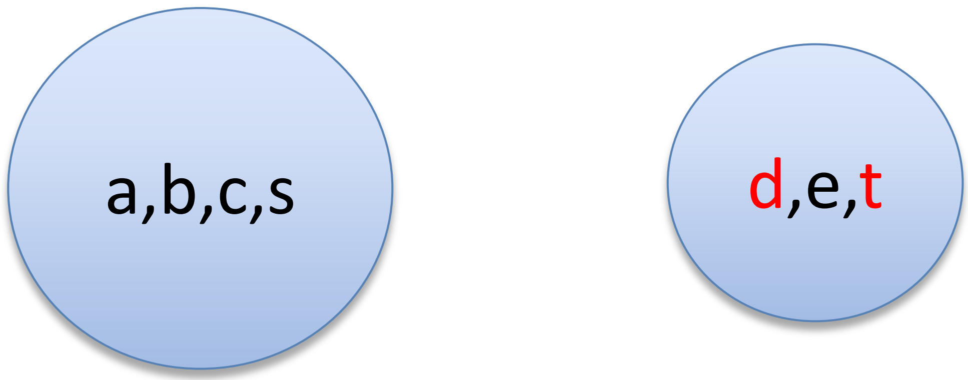
Deciding Equality

a = b, b = c, d = e, b = s, d = t, a ≠ e, a ≠ s



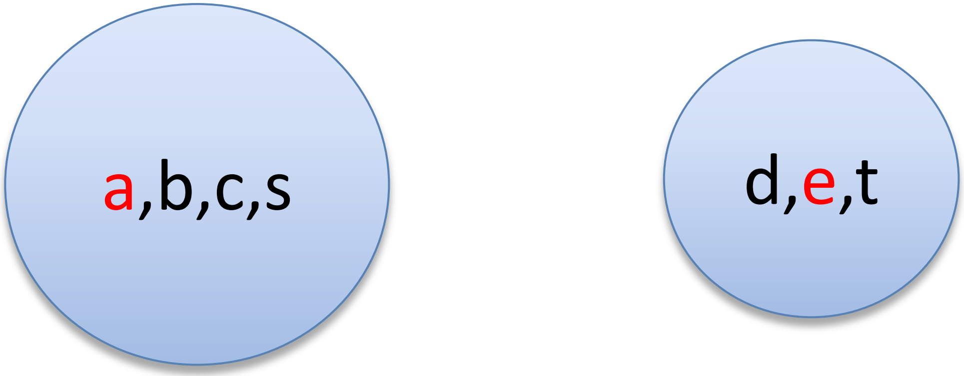
Deciding Equality

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



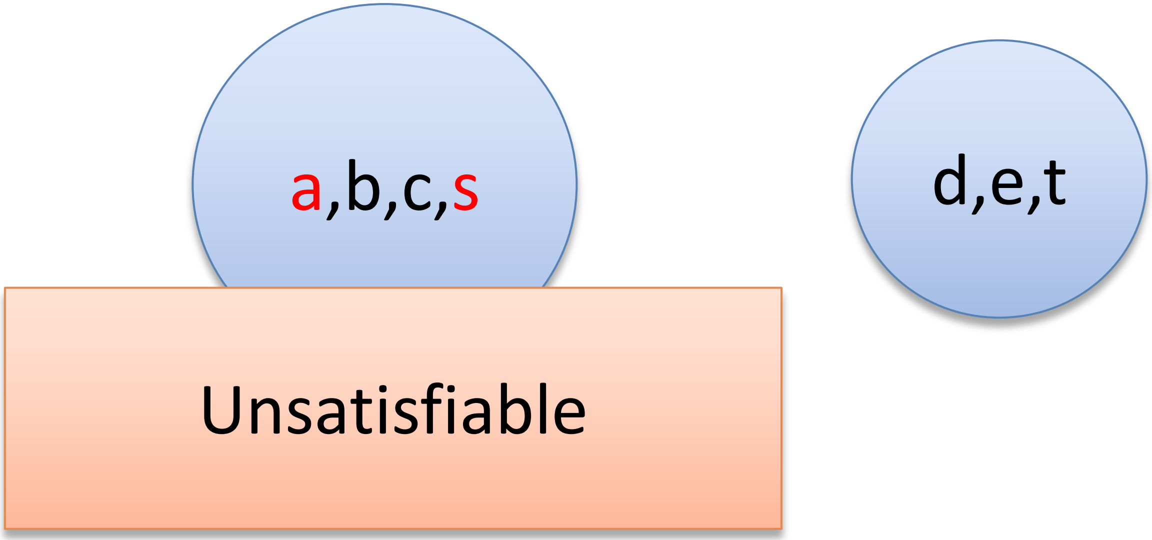
Deciding Equality

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



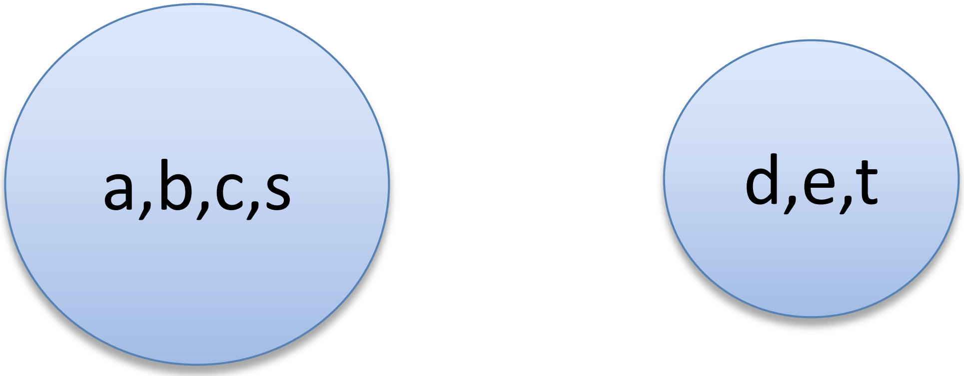
Deciding Equality

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



Deciding Equality

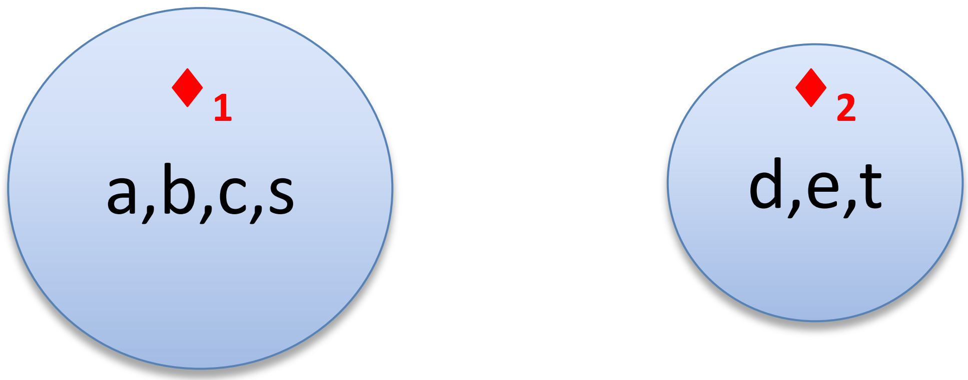
a = b, b = c, d = e, b = s, d = t, a $\neq$ e



Model construction

Deciding Equality

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

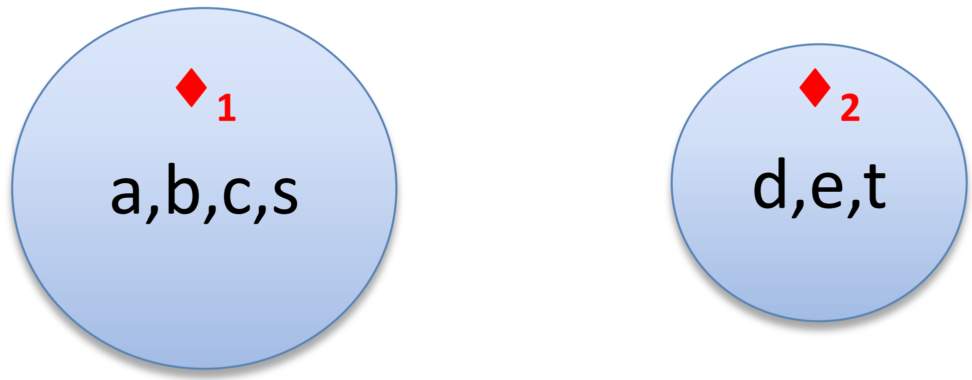


Model construction

$|M| = \{\blacklozenge_1, \blacklozenge_2\}$ (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

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

$\blacklozenge_1$ a,b,c,s

$\blacklozenge_2$ d,e,t

Alternative notation:
$a^M = \blacklozenge_1$

Model construction

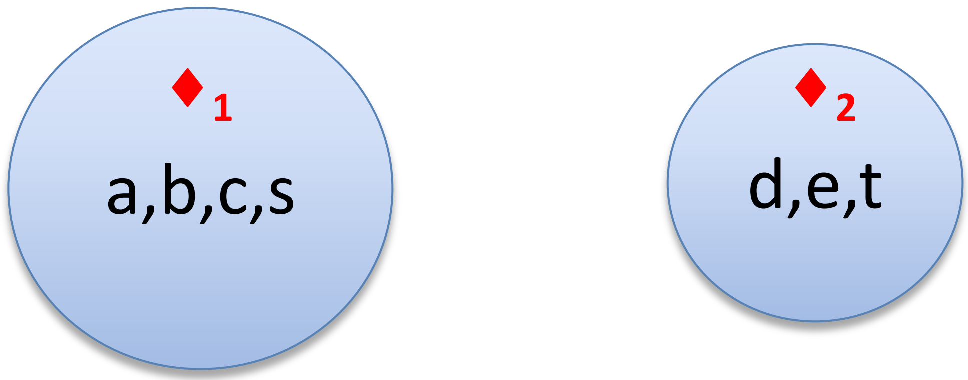
$|M| = \{\blacklozenge_1, \blacklozenge_2\}$ (universe, aka domain)

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

Microsoft Research

Deciding Equality

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



Model construction

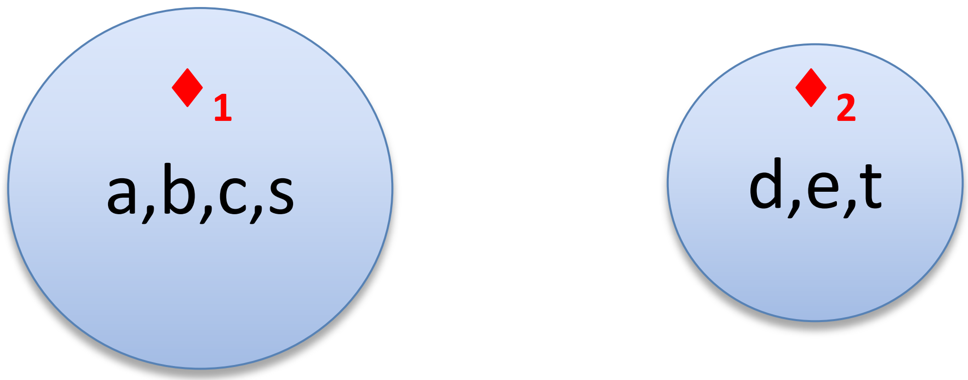
$|M| = \{\blacklozenge_1, \blacklozenge_2\}$ (universe, aka domain)

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

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

Deciding Equality

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



Model construction

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

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

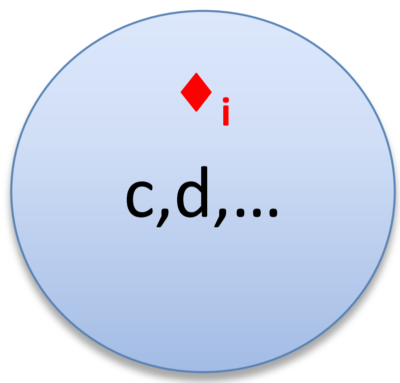
$M(d) = M(e) = M(t) = \blacklozenge_2$

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.

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”.

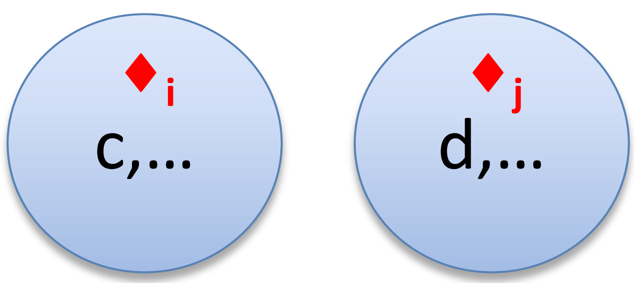


$M(c) = M(d) = \blacklozenge_i$

Microsoft Research

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:

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

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

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

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

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

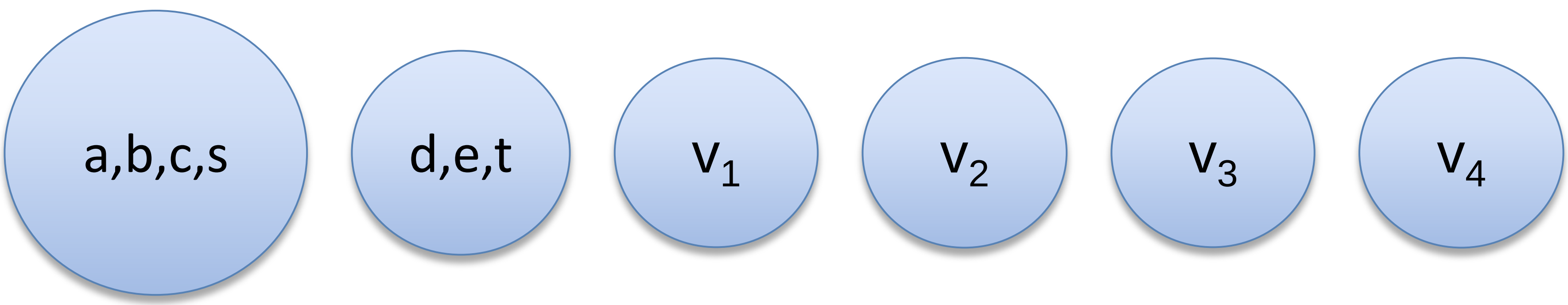
Congruence Rule:

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

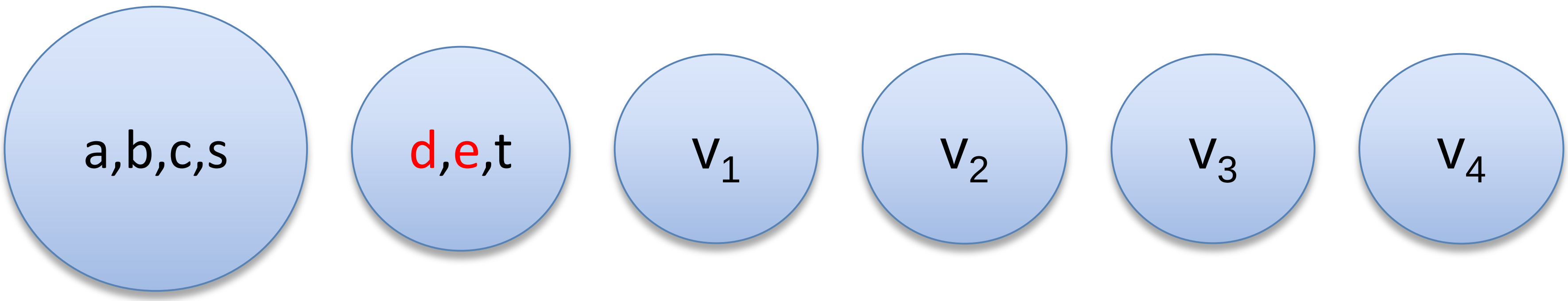
$$x_1 = y_1, \ldots, x_n = y_n \text{ implies } f(x_1, \ldots, x_n) = f(y_1, \ldots, y_n)$$

Microsoft Research

Deciding Equality + (uninterpreted) Functions

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

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



Congruence Rule:

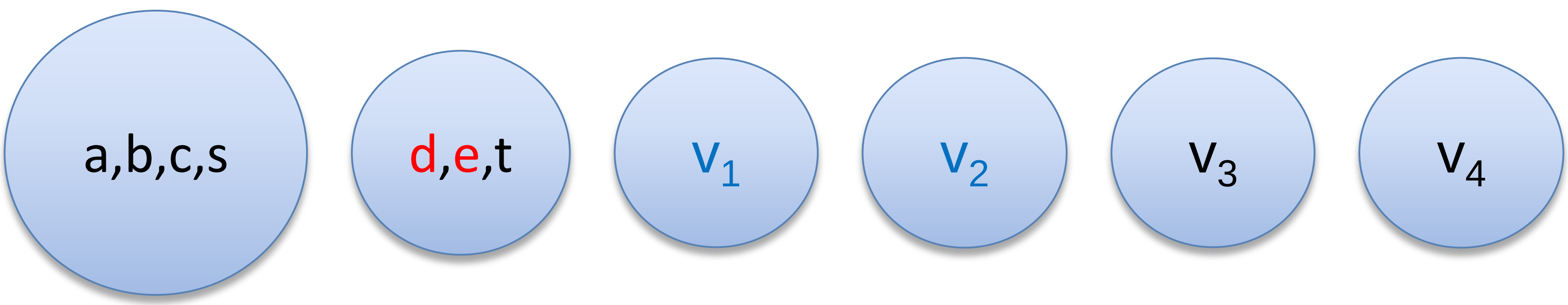
$x_1 = y_1, \ldots, x_n = y_n$ implies $f(x_1, \ldots, x_n) = f(y_1, \ldots, y_n)$

$d = e$ 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:

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

$$d = e \text{ implies } v_1 = v_2$$

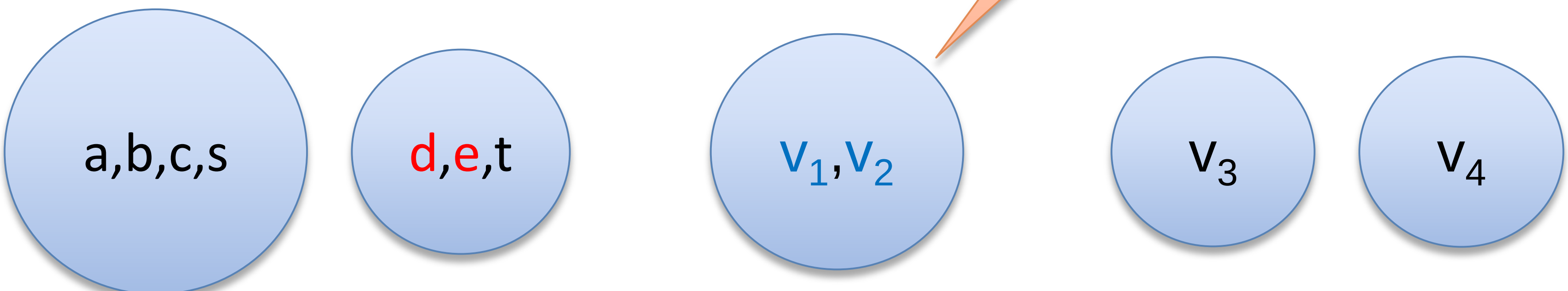
Microsoft Research

Deciding Equality + (uninterpreted) Functions

We say:
v_1 and v_2 are congruent.

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

$v_1 \equiv g(d), v_2 \equiv g(e), v_3 \equiv f(a, v_1) \ldots {}_4 \equiv f(b, v_2)$



Congruence Rule:

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

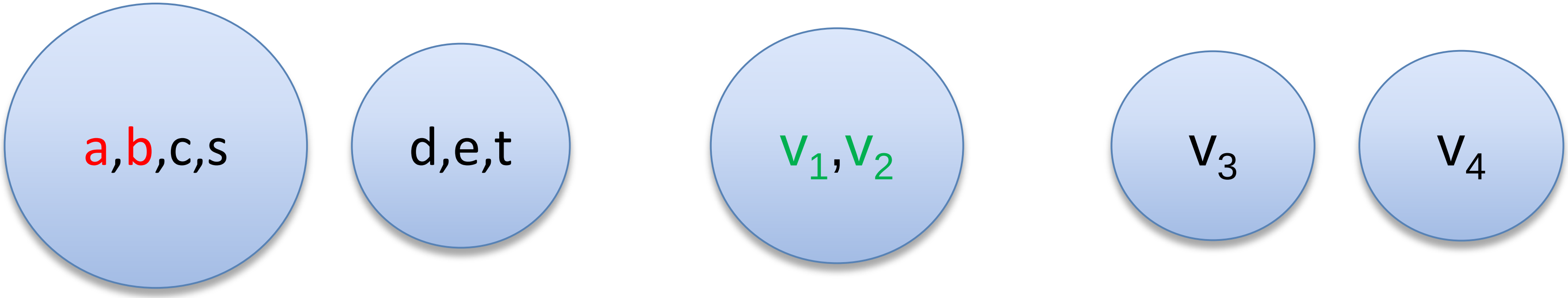
$d = e$ implies $v_1 = v_2$

Microsoft Research

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:

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

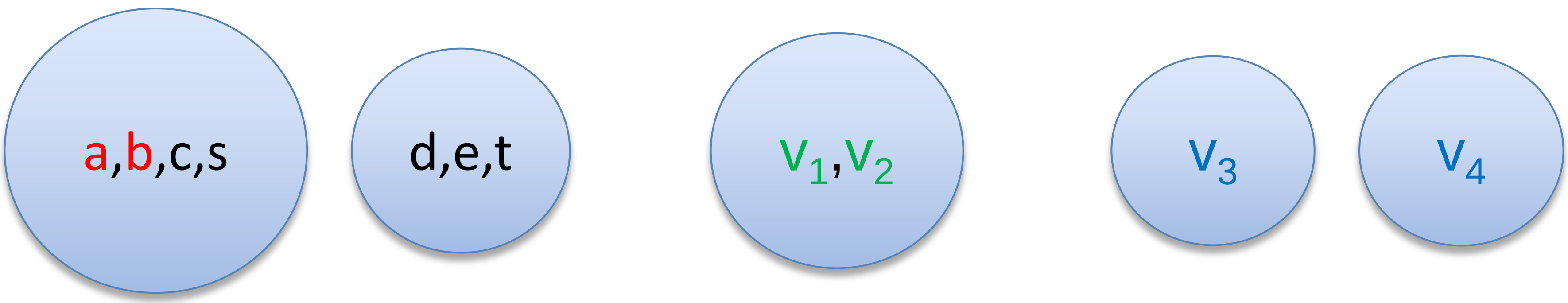
$$a = b,\ v_1 = v_2 \text{ implies } f(a, v_1) = f(b, v_2)$$

Microsoft Research

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

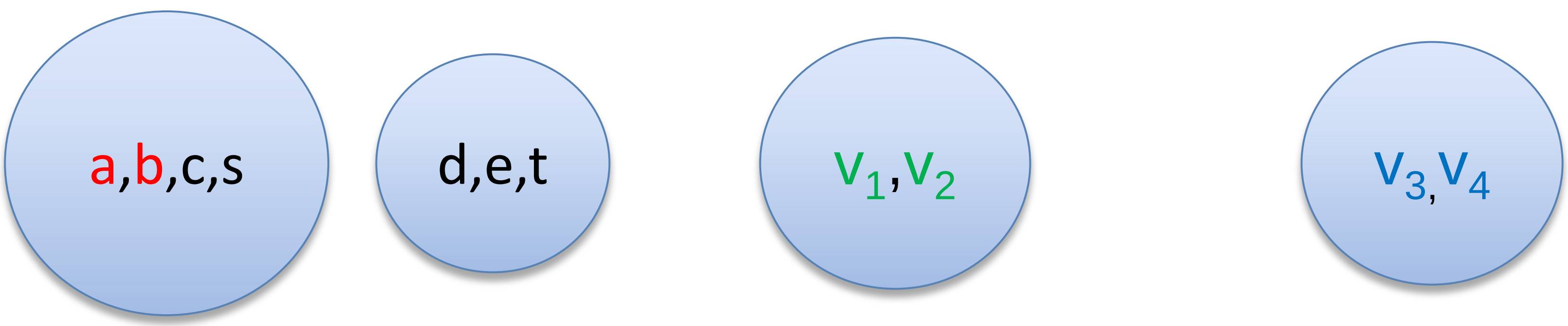
$a = b,\ v_1 = v_2$ implies $v_3 = v_4$

Microsoft Research

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 $v_3 = v_4$

Microsoft Research

Deciding Equality + (uninterpreted) Functions

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

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

a,b,c,s

d,e,t

v_1, v_2

v_3, v_4

Unsatisfiable

Congruence Rule:

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

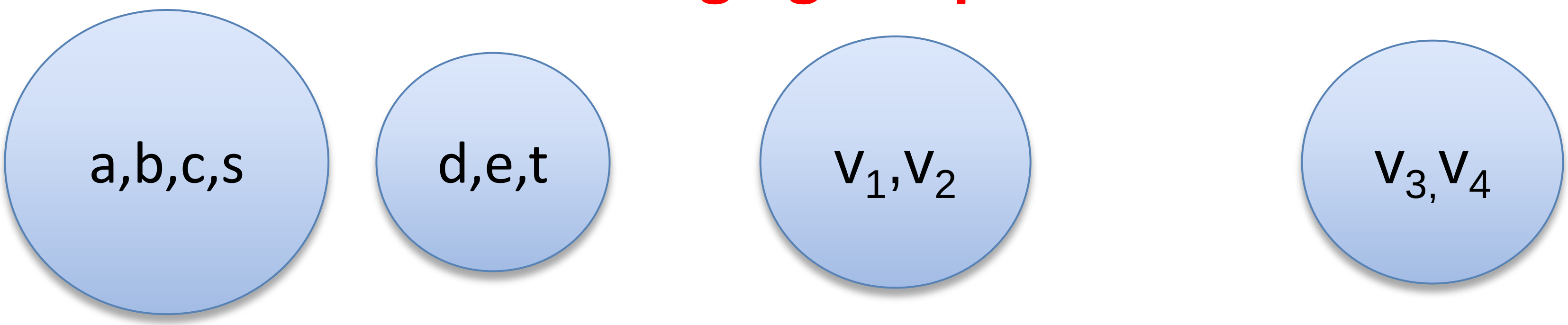
Microsoft Research

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



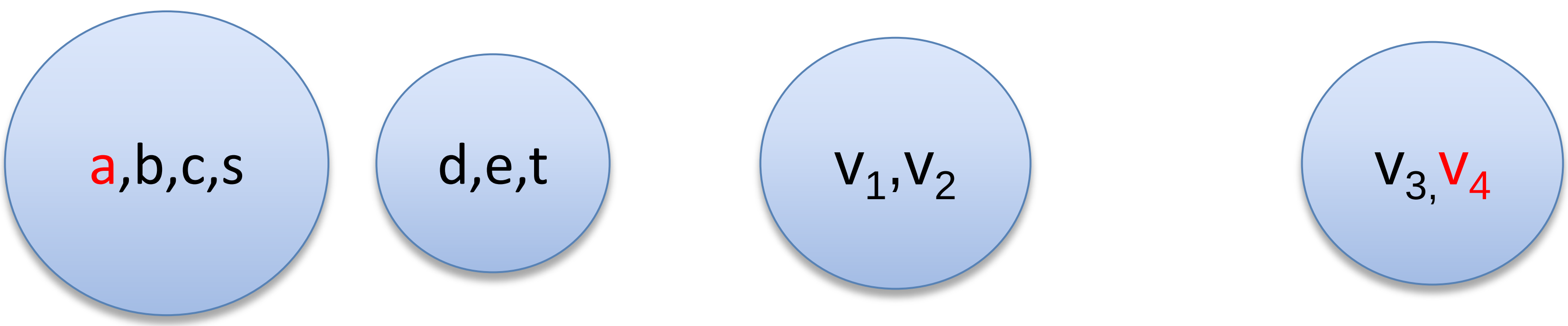
Congruence Rule:

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

Deciding Equality + (uninterpreted) Functions

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

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



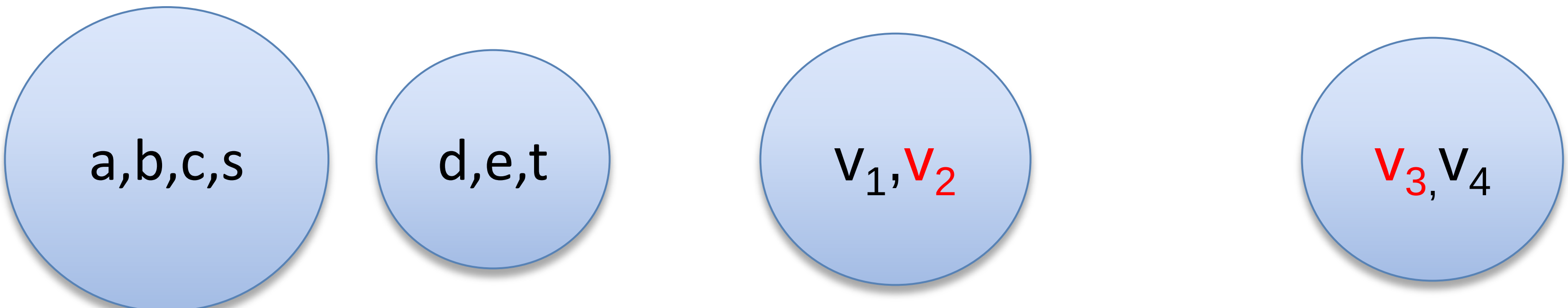
Congruence Rule:

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

Deciding Equality + (uninterpreted) Functions

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

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



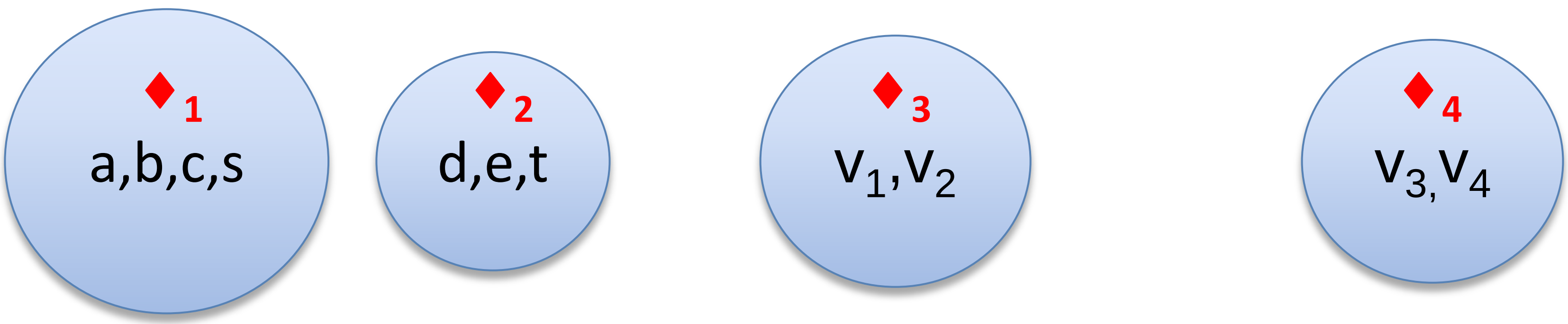
Congruence Rule:

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

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

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

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

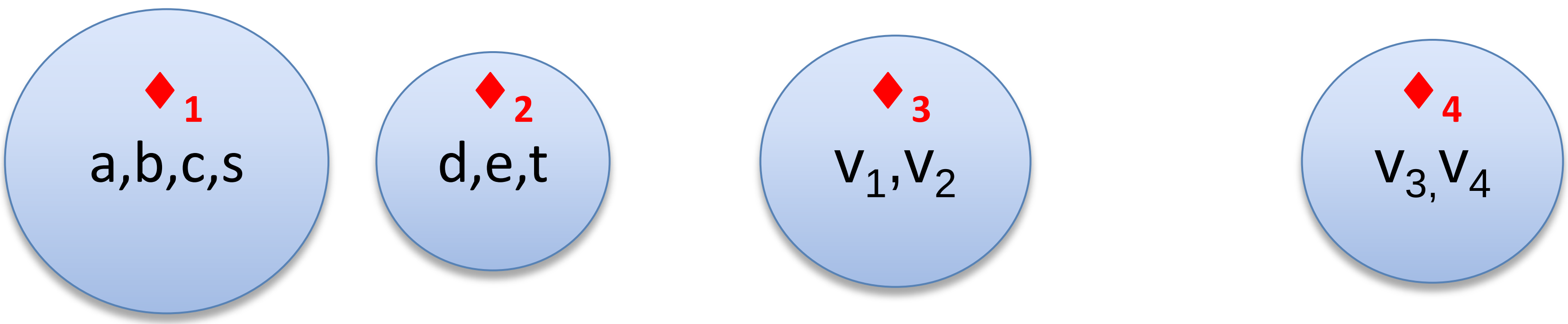
$$M(v_1) = M(v_2) = \blacklozenge_3$$

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

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.

Microsoft Research

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 $v \equiv g(a)$ s.t.

    $M(a) = \blacklozenge_i$

    $M(v) = \blacklozenge_j$

    $= \blacklozenge_k$, otherwise ($\blacklozenge_k$ is an arbitrary element)
  - Is M(g) well-defined?

Microsoft Research

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 $v \equiv g(a)$ s.t.
    $M(a) = \blacklozenge_i$
    $M(v) = \blacklozenge_j$
    $= \blacklozenge_k$, otherwise ($\blacklozenge_k$ 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 $M(g)(\blacklozenge_1) = \blacklozenge_3$?

Microsoft Research

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 $v \equiv g$

    $M(a) = \blacklozenge_i$

    $M(v) = \blacklozenge_j$

    $= \blacklozenge_k$, otherwise ($\blacklozenge_k$ is a ... ary 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 $M(g)(\blacklozenge_1) = \blacklozenge_3$?

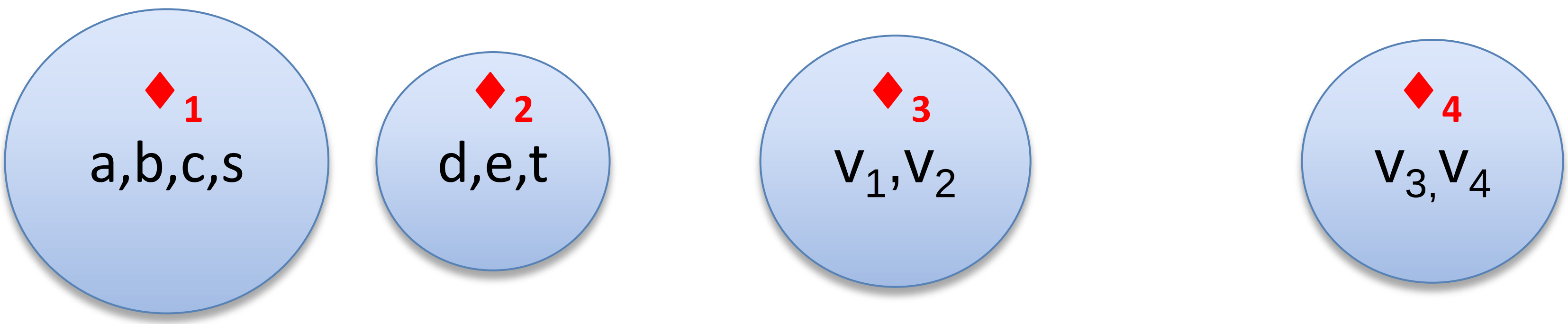
This is impossible because of the congruence rule!
a and b are in the same "ball", then so are v and w

Microsoft Research

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

Microsoft Research

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

Microsoft Research

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

Microsoft Research

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

Microsoft Research

Deciding Equality + (uninterpreted) Functions

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

Deciding Equality + (uninterpreted) Functions

d,e,t — Sets (equivalence classes)

d,e $\cup$ t = d,e,t — Union

a,b,c,s — $a \neq s$ — Membership

Microsoft Research

Deciding Equality + (uninterpreted) Functions

d,e,t

Sets (equivale

Key observation:
The sets are disjoint!

d,e $\cup$ t = d,e,t

Union

a,b,c,s

$a \neq s$

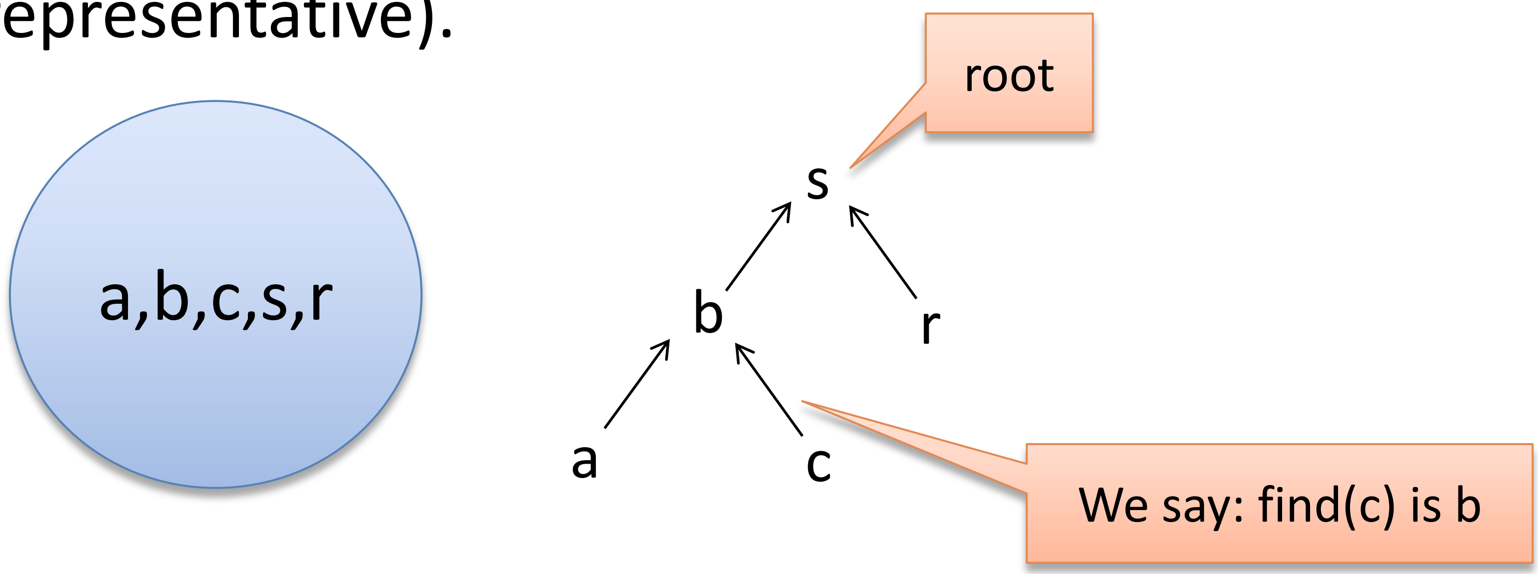
Membership

Microsoft Research

Deciding Equality + (uninterpreted) Functions

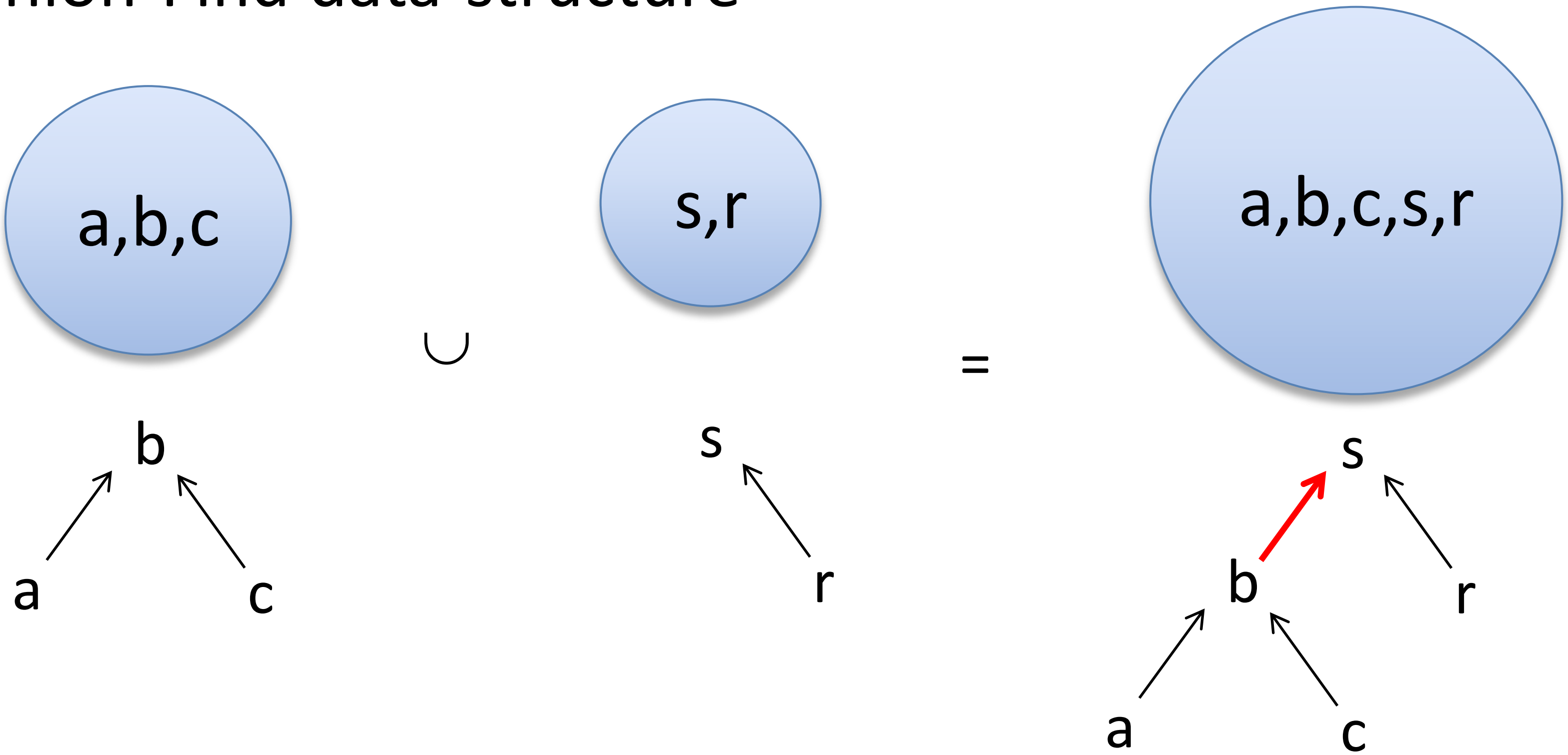
Union-Find data-structure

Every set (equivalence class) has a root element (representative).



Deciding Equality + (uninterpreted) Functions

Union-Find data-structure



Deciding Equality + (uninterpreted) Functions

Tracking the equivalence classes size is important!

$$a_1 \longrightarrow a_2 \quad \cup \quad a_3 \quad = \quad a_1 \longrightarrow a_2 \longrightarrow a_3$$

$$a_1 \longrightarrow a_2 \longrightarrow a_3 \quad \cup \quad a_4 \quad = \quad a_1 \longrightarrow a_2 \longrightarrow a_3 \longrightarrow a_4$$

...

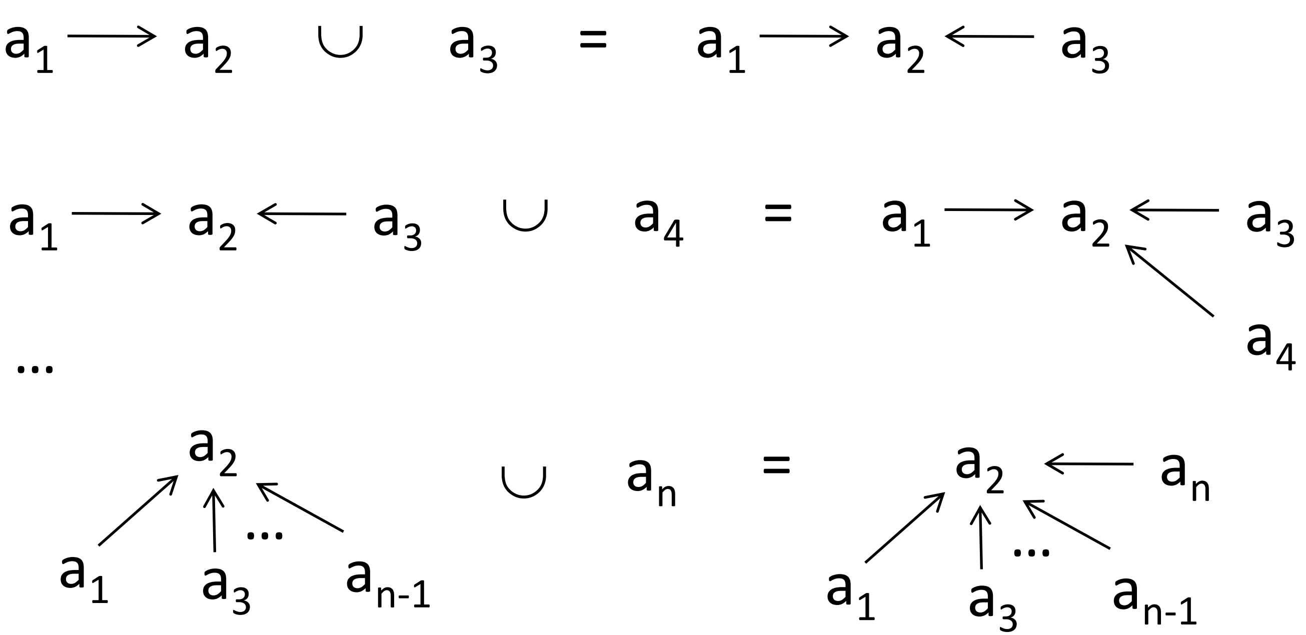
$$a_1 \longrightarrow a_2 \longrightarrow a_3 \longrightarrow \ldots \longrightarrow a_{n-1} \quad \cup \quad a_n \quad =$$

$$a_1 \longrightarrow a_2 \longrightarrow a_3 \longrightarrow \ldots \longrightarrow a_{n-1} \longrightarrow a_n$$

Microsoft Research

Deciding Equality + (uninterpreted) Functions

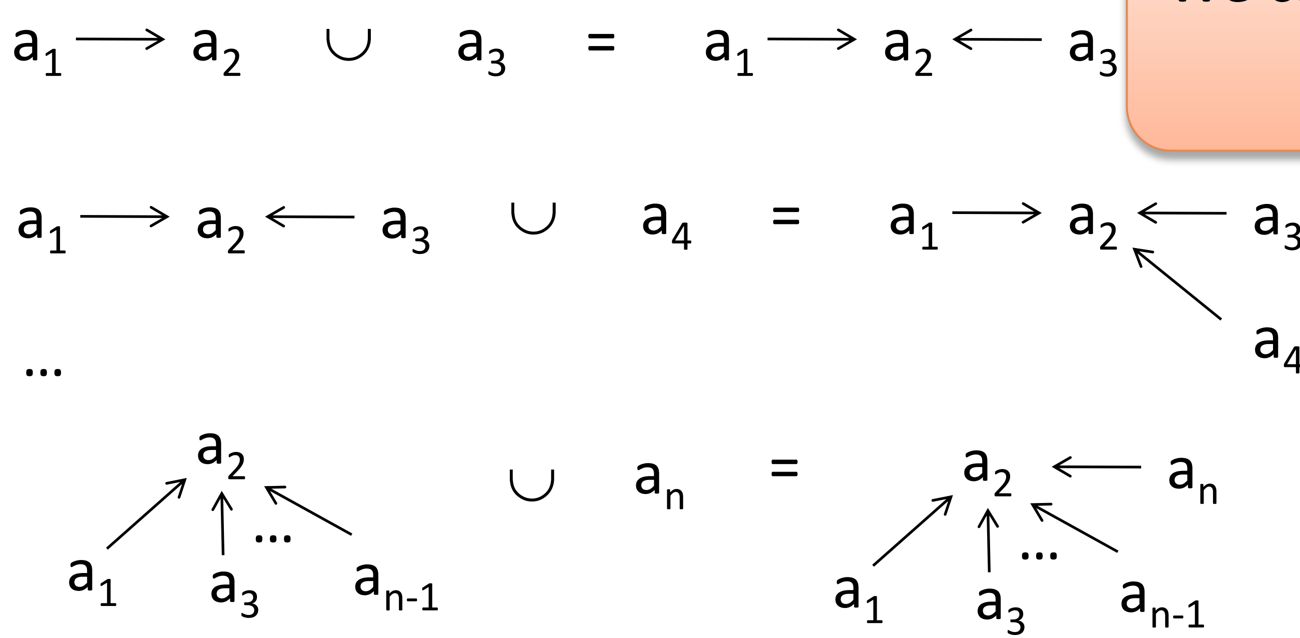
Tracking the equivalence classes size is important!



Deciding Equality + (uninterpreted) Functions

Tracking the equivalence classes size is important!

We can do n merges in O(n log n)



Each constant has two fields: find and size.

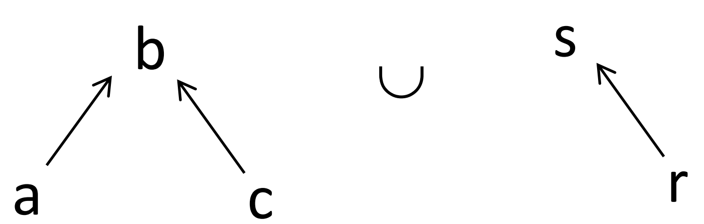
Microsoft Research

Deciding Equality + (uninterpreted) Functions

Implementing the congruence rule.

Occurrences of a constant: we say a occurs in v iff $v \equiv f(\ldots,a,\ldots)$

When we "merge" two equivalence classes we can traverse these occurrences to find new congruences.



Occurrences(b) = { $v_1 \equiv g(b)$, $v_2 \equiv f(a)$ }

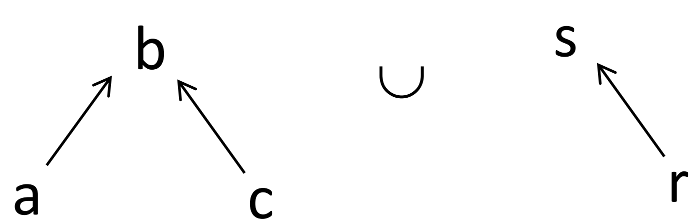
Occurrences(s) = { $v_3 \equiv f(r)$ }

Deciding Equality + (uninterpreted) Functions

Implementing the congruence rule.

Occurrences of a constant: we say a occurs in v iff $v \equiv f(\ldots,a,\ldots)$

When we "merge" two equivalence classes we can traverse these occurrences to find new congruences.



occurrences(b) = { $v_1 \equiv g(b)$, $v_2 \equiv f(a)$ }

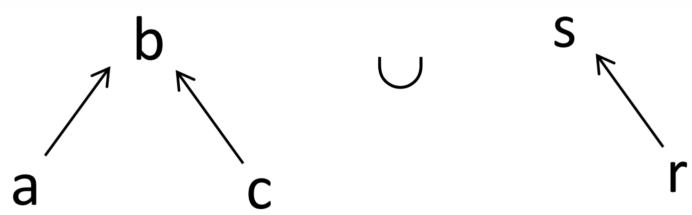
occurrences(s) = { $v_3 \equiv f(r)$ }

Inefficient version:

for each v in occurrences(b)
    for each w in occurrences(s)
        if v and w are congruent
            add (v,w) to todo queue

A queue of pairs that need to be merged.

Deciding Equality + (uninterpreted) Functions



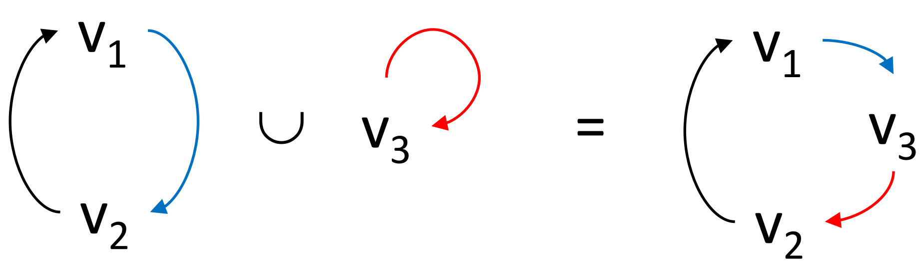
occurrences(b) = { $v_1 \equiv g(b)$, $v_2 \equiv f(a)$ }

occurrences(s) = { $v_3 \equiv f(r)$ }

We also need to merge occurrences(b) with occurrences(s).

This can be done in constant time:

Use circular lists to represent the occurrences. (More later)



Deciding Equality + (uninterpreted) Functions

Avoiding the nested loop:

```
for each v in occurrences(b)
    for each w in occurrences(s)
        …
```

Avoiding the nested loop:

Use a hash table to store the elements $v_1 \equiv f(a_1, \ldots, a_n)$.

Each constant has an identifier (e.g., natural number).

Compute hash code using the identifier of the (equivalence class) roots of the arguments.

$hash(v_1) = hash\text{-}tuple(id(f), id(root(a_1)), \ldots, id(root(a_n)))$

Microsoft Research

Deciding Equality + (uninterpreted) Functions

Avoiding the nested loop:

for each v in occurrences(b)

    for each w in occurrences(s)

        …

Avoiding the nested loop:

Use a hash table to … , …, a_n).

Each constant has a … umber).

Compute hash cod… (equivalence class) roots of the argu… .

hash-tuple can be the Jenkin's hash function for strings.

Just adding the ids produces a very bad hash-code!

$hash(v_1)$ = hash-tuple(id(f), id(root(a_1)), …, id(root(a_n)))

Microsoft Research

Deciding Equality + (uninterpreted) Functions

Efficient implementation of the congruence rule.

Merging the equivalences classes with roots: a_1 and a_2

Assume a_2 is smaller than a_1

Before merging the equivalence classes: a_1 and a_2

for each v in occurrences(a_2)

    remove v from the hash table   (its hashcode will change)

After merging the equivalence classes: a_1 and a_2

for each v in occurrences(a_2)

    if there is w congruent to v in the hash-table

        add (v,w) to todo queue

    else add v to hash-table

Microsoft Research

Deciding Equality + (uninterpreted) Functi

Trick:
Use dynamic arrays to represent the occurrences

Efficient implementation of the congru

Merging the equivalences classes with roo $_1$ and a_2

Assume a_2 is smaller than a_1

Before merging the equivalence classes: a_1 and a_2

for each v in occurrences(a_2)

    remove v from the hash table   (its hashcode will change)

After merging the equivalence classes: a_1 and a_2

for each v in occurrences(a_2)

    if there is w congruent to v in the hash-table

        add (v,w) to todo queue

    else add v to hash-table

        add v to occurrences(a_1)

Deciding Equality + (uninterpreted) Functions

The efficient version is not optimal (in theory).

Problem: we may have $v = f(a_1, \ldots, a_n)$ with "huge" n.

Solution: currying

Use only binary functions, and represent $f(a_1, a_2, a_3, a_4)$ as $f(a_1, h(a_2, h(a_3, a_4)))$

This is not necessary in practice, since the n above is small.

Deciding Equality + (uninterpreted) Functions

Each constant has now three fields:
find, size, and occurrences.

We also has use a hash-table for implementing the congruence rule.

We will need many more improvements!

Case Analysis

Many verification/analysis problems require:

case-analysis

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

Microsoft Research

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

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

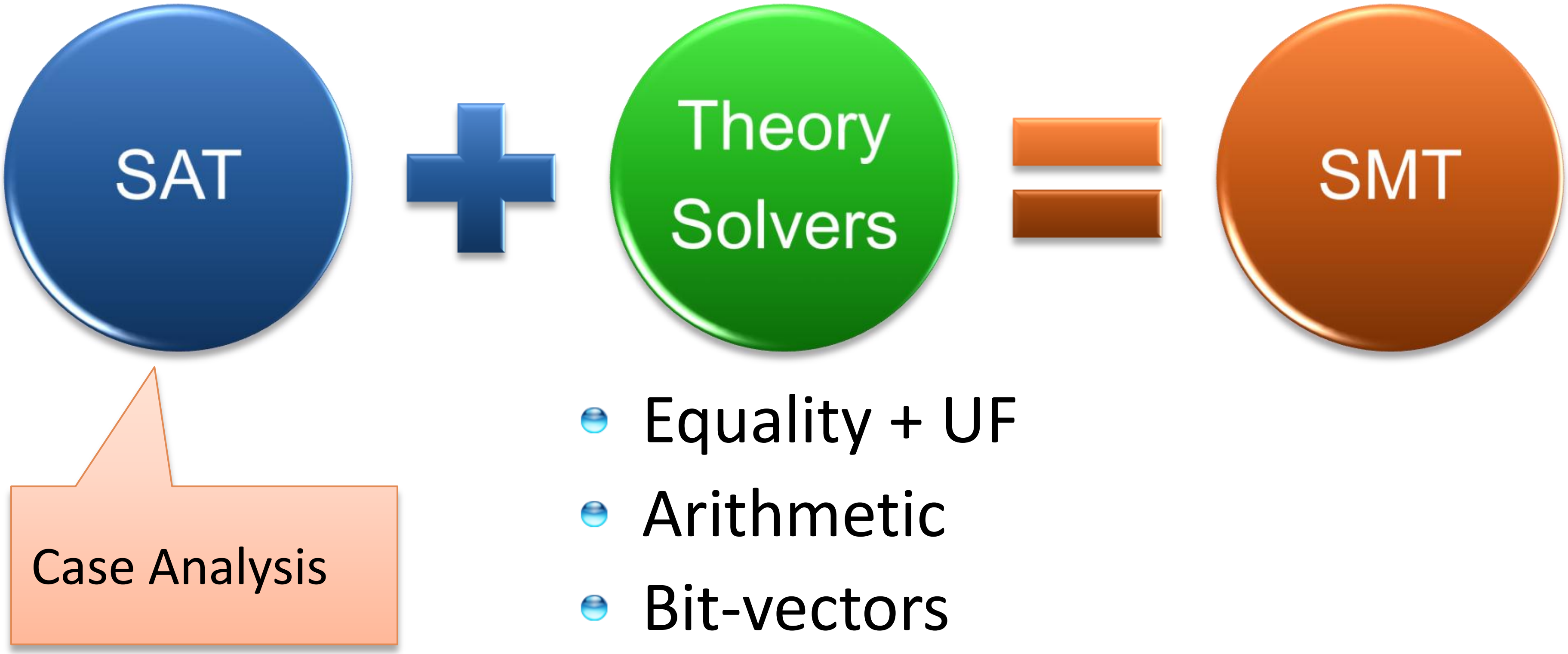
Naïve Solution: Convert to DNF

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

Too Inefficient!
(exponential blowup)

Microsoft Research

SMT : Basic Architecture



- Equality + UF
- Arithmetic
- Bit-vectors
- …

DPLL (abstract view)

M | F

Partial model

Set of clauses

DPLL (abstract view)

Guessing

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

↓

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

DPLL (abstract view)

Deducing

$$p \;|\; p \vee q, \neg p \vee s$$

$$\Downarrow$$

$$p, s \;|\; p \vee q, \neg p \vee s$$

DPLL (abstract view)

Backtracking

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

↓

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

Microsoft Research

Modern DPLL

- Efficient indexing (two-watch literal)
- Non-chronological backtracking (backjumping)
- Lemma learning

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

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

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

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

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

$\neg p_1 \lor \neg p_2 \lor \neg p_4$

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

Abstract (aka "naming" atoms)

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

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

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
$\neg p_1 \lor \neg p_2 \lor \neg p_4$

SAT + Theory solvers

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

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

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

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

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

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.

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

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

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

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

Imprecise Lemma

SAT + Theory solvers

Theory Propagation

It is the SMT equivalent of unit propagation.

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

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

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

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.

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

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

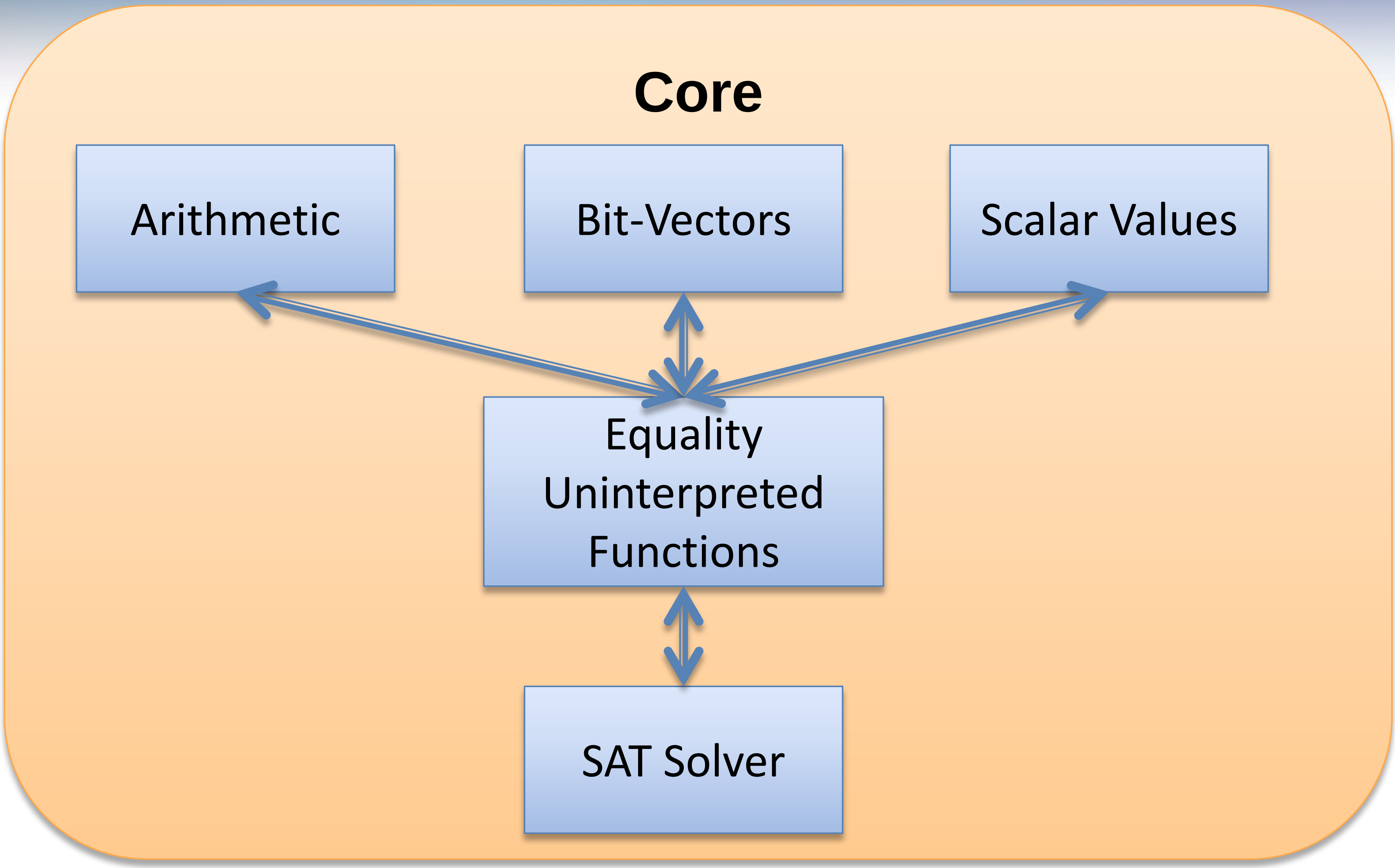
$p_1,\ p_2 \mid p_1,\ p_2, (p_3 \vee p_4), (p_5 \vee \neg p_4)$

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

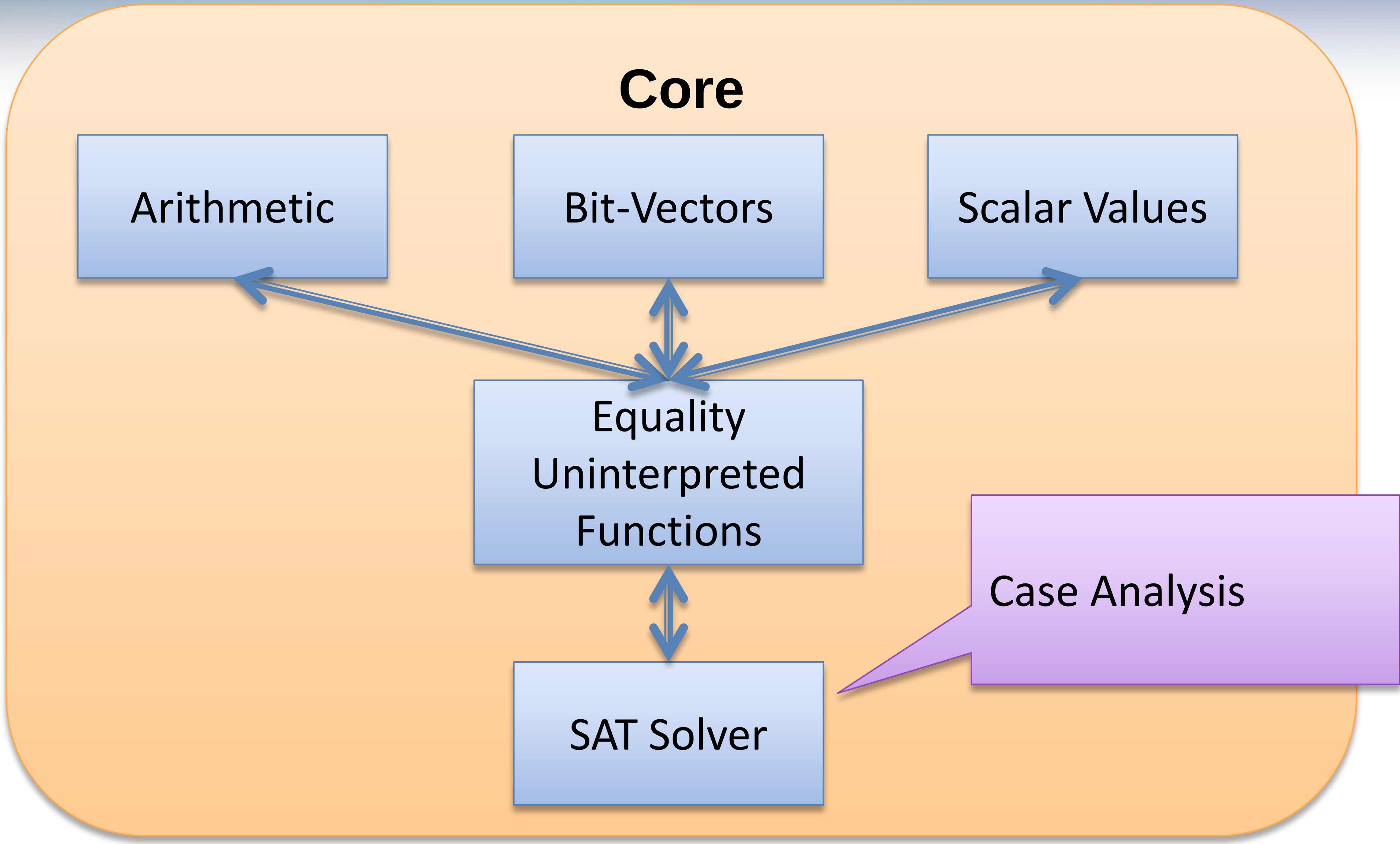
$p_1,\ p_2,\ \neg p_4 \mid p_1,\ p_2, (p_3 \vee p_4), (p_5 \vee \neg p_4)$

Tradeoff between precision × performance.

An Architecture: the core



An Architecture: the core



An Architecture: the core