ECE750T-28:
Computer-aided Reasoning for Software Engineering

Lecture 11: Theory of Equality with Uninterpreted Functions

Vijay Ganesh
(Original notes from Isil Dillig)

Review

- Previous lecture: talked about signature and axioms of $T_=$

$$\Sigma_= : \; \{=, \; a, \; b, \; c, \; \ldots, \; f, \; g, \; h, \; \ldots, \; p, \; q, \; r, \; \ldots\}$$

Review

- Previous lecture: talked about signature and axioms of $T_=$

$$\Sigma_= : \{=,\ a,\ b,\ c,\ \ldots,\ f,\ g,\ h,\ \ldots,\ p,\ q,\ r,\ \ldots\}$$

- Axioms:
  1. $\forall x.\ x = x$ (reflexivity)
  2. $\forall x, y.\ x = y \ \rightarrow\ y = x$ (symmetry)
  3. $\forall x, y, z.\ x = y \ \wedge\ y = z \ \rightarrow\ x = z$ (transitivity)
  4. $\forall x_1, \ldots, x_n, y_1, \ldots, y_n.\ \bigwedge_i x_i = y_i \ \rightarrow\ f(x_1, \ldots, x_n) = f(y_1, \ldots, y_n)$ (congruence)
  5. for each positive integer n and n-ary predicate symbol p,
     $$\forall x_1, \ldots, x_n, y_1, \ldots, y_n.\ \bigwedge_i x_i = y_i \ \rightarrow\ (p(x_1, \ldots, x_n) \ \leftrightarrow\ p(y_1, \ldots, y_n))$$ (equivalence)

Overview

- Today: look at decision procedures for deciding satisfiability in the quantifier-free fragment of $T_=$

Overview

- Today: look at decision procedures for deciding satisfiability in the quantifier-free fragment of $T_=$
- However, our decision procedure has two "restrictions":

Overview

- Today: look at decision procedures for deciding satisfiability in the quantifier-free fragment of $T_=$
- However, our decision procedure has two "restrictions":
  - formulas consist of conjunctions of literals

Overview

- Today: look at decision procedures for deciding satisfiability in the quantifier-free fragment of $T_=$
- However, our decision procedure has two "restrictions":
  - formulas consist of conjunctions of literals
  - we'll allow functions, but no predicates

Overview

- Today: look at decision procedures for deciding satisfiability in the quantifier-free fragment of $T_=$
- However, our decision procedure has two "restrictions":
  - formulas consist of conjunctions of literals
  - we'll allow functions, but no predicates
- However, these "restrictions" are not real restrictions

Overview

- Today: look at decision procedures for deciding satisfiability in the quantifier-free fragment of $T_=$
- However, our decision procedure has two "restrictions":
  - formulas consist of conjunctions of literals
  - we'll allow functions, but no predicates
- However, these "restrictions" are not real restrictions
- For formulas with disjunctions, can convert to DNF and check each clause separately (will consider efficient methods later)

Overview

- Today: look at decision procedures for deciding satisfiability in the quantifier-free fragment of $T_=$
- However, our decision procedure has two "restrictions":
  - formulas consist of conjunctions of literals
  - we'll allow functions, but no predicates
- However, these "restrictions" are not real restrictions
- For formulas with disjunctions, can convert to DNF and check each clause separately (will consider efficient methods later)
- Furthermore, any formula containing predicates can be converted to equisatisfiable formula containing only functions!

Eliminating Predicates

- Simple transformation yields equisatisfiable formula with only functions

Eliminating Predicates

- Simple transformation yields equisatisfiable formula with only functions
- The trick: For each relation constant p:
  1. introduce a fresh function constant f_p

Eliminating Predicates

- Simple transformation yields equisatisfiable formula with only functions
- The trick: For each relation constant p:
  1. introduce a fresh function constant f_p
  2. rewrite $p(x_1, \ldots, x_n)$ as $f_p(x_1, \ldots, x_n) = t$

  where t is a fresh object constant

Eliminating Predicates

- Simple transformation yields equisatisfiable formula with only functions
- The trick: For each relation constant p:
  1. introduce a fresh function constant f_p
  2. rewrite $p(x_1, \ldots, x_n)$ as $f_p(x_1, \ldots, x_n) = t$

  where t is a fresh object constant
- Example: How do we transform $x = y \rightarrow (p(x) \leftrightarrow p(y))$ to equisat formula?

Eliminating Predicates

- Simple transformation yields equisatisfiable formula with only functions
- The trick: For each relation constant p:
  1. introduce a fresh function constant f_p
  2. rewrite $p(x_1, \ldots, x_n)$ as $f_p(x_1, \ldots, x_n) = t$

  where t is a fresh object constant
- Example: How do we transform $x = y \rightarrow (p(x) \leftrightarrow p(y))$ to equisat formula? $x = y \rightarrow (f_p(x) = t \leftrightarrow f_p(y) = t)$

$T_=$ without Predicates

- Signature without predicates:

$$\Sigma_= : \{=,\ a,\ b,\ c,\ \ldots,\ f,\ g,\ h,\ \ldots\}$$

$T_=$ without Predicates

- Signature without predicates:

$$\Sigma_= : \ \{=, \ a, \ b, \ c, \ \ldots, \ f, \ g, \ h, \ \ldots\}$$

- Axioms:
  1. $\forall x. \ x = x$ (reflexivity)
  2. $\forall x, y. \ x = y \ \rightarrow \ y = x$ (symmetry)
  3. $\forall x, y, z. \ x = y \ \wedge \ y = z \ \rightarrow \ x = z$ (transitivity)
  4. $\forall x_1, \ldots, x_n, y_1, \ldots, y_n. \ \bigwedge_i x_i = y_i \ \rightarrow \ f(x_1, \ldots, x_n) = f(y_1, \ldots, y_n)$ (congruence)

Examples

- Let's consider some examples

Examples

- Let's consider some examples
- Is the formula $x = y \land f(x) \neq f(y)$ sat, unsat, valid?

Examples

- Let's consider some examples
- Is the formula $x = y \wedge f(x) \neq f(y)$ sat, unsat, valid? unsat

Examples

- Let's consider some examples
- Is the formula $x = y \wedge f(x) \neq f(y)$ sat, unsat, valid? unsat
- What about $x \neq y \wedge f(x) = f(y)$?

Examples

- Let's consider some examples
- Is the formula $x = y \wedge f(x) \neq f(y)$ sat, unsat, valid? unsat
- What about $x \neq y \wedge f(x) = f(y)$? sat

Examples

- Let's consider some examples
- Is the formula $x = y \land f(x) \neq f(y)$ sat, unsat, valid? unsat
- What about $x \neq y \land f(x) = f(y)$? sat
- What about $x = g(y, z) \rightarrow f(x) = f(g(y, z))$?

Examples

- Let's consider some examples
- Is the formula $x = y \land f(x) \neq f(y)$ sat, unsat, valid? unsat
- What about $x \neq y \land f(x) = f(y)$? sat
- What about $x = g(y, z) \rightarrow f(x) = f(g(y, z))$? valid

Examples

- Let's consider some examples
- Is the formula $x = y \wedge f(x) \neq f(y)$ sat, unsat, valid? unsat
- What about $x \neq y \wedge f(x) = f(y)$? sat
- What about $x = g(y, z) \rightarrow f(x) = f(g(y, z))$? valid
- What about $f(a) = a \wedge f(f(a)) \neq a$?

Examples

- Let's consider some examples
- Is the formula $x = y \land f(x) \neq f(y)$ sat, unsat, valid? unsat
- What about $x \neq y \land f(x) = f(y)$? sat
- What about $x = g(y, z) \rightarrow f(x) = f(g(y, z))$? valid
- What about $f(a) = a \land f(f(a)) \neq a$? unsat

Example, cont.

- What about $f(f(f(a))) = a \land f(f(f(f(f(a))))) = a \land f(a) \neq a$?

Example, cont.

- What about $f(f(f(a))) = a \land f(f(f(f(f(a))))) = a \land f(a) \neq a$? unsat

- Reasoning: Substitute a for $f(f(f(a)))$ in second equality, this yields: $f(f(a)) = a$

Example, cont.

- What about $f(f(f(a))) = a \land f(f(f(f(f(a))))) = a \land f(a) \neq a$? unsat
- Reasoning: Substitute a for $f(f(f(a)))$ in second equality, this yields: $f(f(a)) = a$
- Since $f(f(a)) = a$, by congruence $f(f(f(a))) = f(a)$

Example, cont.

- What about $f(f(f(a))) = a \wedge f(f(f(f(f(a))))) = a \wedge f(a) \neq a$? unsat
- Reasoning: Substitute a for $f(f(f(a)))$ in second equality, this yields: $f(f(a)) = a$
- Since $f(f(a)) = a$, by congruence $f(f(f(a))) = f(a)$
- By first equality, we have $f(a) = a \Rightarrow$ contradiction!

Equivalence Relations

- Decision procedure for theory of equality known as congruence closure algorithm

Equivalence Relations

- Decision procedure for theory of equality known as congruence closure algorithm
- But need to understand what congruence closure is first $\Rightarrow$ new terminology and concepts

Equivalence Relations

- Decision procedure for theory of equality known as congruence closure algorithm
- But need to understand what congruence closure is first $\Rightarrow$ new terminology and concepts
- A binary relation R over a set S is an equivalence relation if

Equivalence Relations

- Decision procedure for theory of equality known as congruence closure algorithm
- But need to understand what congruence closure is first $\Rightarrow$ new terminology and concepts
- A binary relation R over a set S is an equivalence relation if
  1. reflexive: $\forall s \in S.\ sRs$

Equivalence Relations

- Decision procedure for theory of equality known as congruence closure algorithm
- But need to understand what congruence closure is first $\Rightarrow$ new terminology and concepts
- A binary relation R over a set S is an equivalence relation if
  1. reflexive: $\forall s \in S.\ sRs$
  2. symmetric: $\forall s_1, s_2 \in S.\ s_1 R s_2 \ \rightarrow \ s_2 R s_1$;

Equivalence Relations

- Decision procedure for theory of equality known as congruence closure algorithm
- But need to understand what congruence closure is first $\Rightarrow$ new terminology and concepts
- A binary relation R over a set S is an equivalence relation if
  1. reflexive: $\forall s \in S.\ sRs$
  2. symmetric: $\forall s_1, s_2 \in S.\ s_1 R s_2 \ \rightarrow\ s_2 R s_1$;
  3. transitive: $\forall s_1, s_2, s_3 \in S.\ s_1 R s_2 \ \wedge\ s_2 R s_3 \ \rightarrow\ s_1 R s_3$.

Equivalence and Congruence Relations

- Equality predicate $=$ is equivalence relation over real numbers

Equivalence and Congruence Relations

- Equality predicate $=$ is equivalence relation over real numbers
- The relation "has same birthday as" is an equivalence relation over set of people

Equivalence and Congruence Relations

- Equality predicate $=$ is equivalence relation over real numbers
- The relation "has same birthday as" is an equivalence relation over set of people
- The relation $\equiv_2$ is equivalence relation over $\mathbb{Z}$

Equivalence and Congruence Relations

- Equality predicate $=$ is equivalence relation over real numbers
- The relation "has same birthday as" is an equivalence relation over set of people
- The relation $\equiv_2$ is equivalence relation over $\mathbb{Z}$
- A relation R is congruence relation over set S if it is an equivalence relation and for every n'ary function f:

$$\forall \vec{s}, \vec{t}.\ \bigwedge_{i=1}^{n} s_i R t_i \ \rightarrow\ f(\vec{s})\ R\ f(\vec{t})\ .$$

Equivalence and Congruence Classes

- For a given equivalence relation over S, every member of S belongs to an equivalence class

Equivalence and Congruence Classes

- For a given equivalence relation over S, every member of S belongs to an equivalence class
- The equivalence class of $s \in S$ under R is the set:

$$[s]_R \stackrel{\text{def}}{=} \{s' \in S \ : \ sRs'\} \ .$$

Equivalence and Congruence Classes

- For a given equivalence relation over S, every member of S belongs to an equivalence class
- The equivalence class of $s \in S$ under R is the set:

$$[s]_R \stackrel{\text{def}}{=} \{s' \in S \ : \ sRs'\} \ .$$

- If R is a congruence relation, then this set is called congruence class

Equivalence and Congruence Classes

- For a given equivalence relation over S, every member of S belongs to an equivalence class
- The equivalence class of $s \in S$ under R is the set:

$$[s]_R \stackrel{\text{def}}{=} \{s' \in S \ : \ sRs'\} \ .$$

- If R is a congruence relation, then this set is called congruence class
- Example: What is the equivalence class of 1 under $\equiv_2$?

Equivalence and Congruence Classes

- For a given equivalence relation over S, every member of S belongs to an equivalence class
- The equivalence class of $s \in S$ under R is the set:

$$[s]_R \stackrel{\text{def}}{=} \{s' \in S \ : \ sRs'\} \ .$$

- If R is a congruence relation, then this set is called congruence class
- Example: What is the equivalence class of 1 under $\equiv_2$? odd numbers

Equivalence and Congruence Classes

- For a given equivalence relation over S, every member of S belongs to an equivalence class
- The equivalence class of $s \in S$ under R is the set:

$$[s]_R \stackrel{\text{def}}{=} \{s' \in S \ : \ sRs'\} \ .$$

- If R is a congruence relation, then this set is called congruence class
- Example: What is the equivalence class of 1 under $\equiv_2$? odd numbers
- What is the equivalence class of 6 under $\equiv_3$?

Equivalence and Congruence Classes

- For a given equivalence relation over S, every member of S belongs to an equivalence class
- The equivalence class of $s \in S$ under R is the set:

$$[s]_R \stackrel{\text{def}}{=} \{s' \in S \ : \ sRs'\} \ .$$

- If R is a congruence relation, then this set is called congruence class
- Example: What is the equivalence class of 1 under $\equiv_2$? odd numbers
- What is the equivalence class of 6 under $\equiv_3$? multiples of 3

Relation Refinements

- A binary relation R_1 is a refinement of another binary relation R_2, written $R_1 \prec R_2$, if

$$\forall s_1, s_2 \in S.\ s_1 R_1 s_2 \ \rightarrow\ s_1 R_2 s_2\ .$$

Relation Refinements

- A binary relation R_1 is a refinement of another binary relation R_2, written $R_1 \prec R_2$, if

$$\forall s_1, s_2 \in S.\ s_1 R_1 s_2 \ \rightarrow \ s_1 R_2 s_2 \ .$$

- Example 1: Consider set $S = \{a, b\}$, and relations $R_1 = \{\langle a, b\rangle\}$ and $R_2 = \{\langle a, b\rangle, \langle b, b\rangle\}$

Relation Refinements

- A binary relation R_1 is a refinement of another binary relation R_2, written $R_1 \prec R_2$, if
$$\forall s_1, s_2 \in S.\ s_1 R_1 s_2 \ \rightarrow\ s_1 R_2 s_2\ .$$
- Example 1: Consider set $S = \{a, b\}$, and relations $R_1 = \{\langle a, b\rangle\}$ and $R_2 = \{\langle a, b\rangle, \langle b, b\rangle\}$
- Do either of these hold? $R_1 \prec R_2$ or $R_2 \prec R_1$?

Relation Refinements

- A binary relation R_1 is a refinement of another binary relation R_2, written $R_1 \prec R_2$, if
$$\forall s_1, s_2 \in S.\ s_1 R_1 s_2 \ \rightarrow \ s_1 R_2 s_2 \ .$$
- Example 1: Consider set $S = \{a, b\}$, and relations $R_1 = \{\langle a, b \rangle\}$ and $R_2 = \{\langle a, b \rangle, \langle b, b \rangle\}$
- Do either of these hold? $R_1 \prec R_2$ or $R_2 \prec R_1$? $R_1 \prec R_2$

Relation Refinements

- A binary relation R_1 is a refinement of another binary relation R_2, written $R_1 \prec R_2$, if

$$\forall s_1, s_2 \in S.\; s_1 R_1 s_2 \;\rightarrow\; s_1 R_2 s_2 \;.$$

- Example 1: Consider set $S = \{a, b\}$, and relations $R_1 = \{\langle a, b\rangle\}$ and $R_2 = \{\langle a, b\rangle, \langle b, b\rangle\}$

- Do either of these hold? $R_1 \prec R_2$ or $R_2 \prec R_1$? $R_1 \prec R_2$

- Example 2: Consider set $\mathbb{Z}$ and the relations:
$R_1 : \{xR_1y \;:\; x \text{ mod } 2 = y \text{ mod } 2\}$ $\quad R_2 : \{xR_2y \;:\; x \text{ mod } 4 = y \text{ mod } 4\}$

Relation Refinements

- A binary relation R_1 is a refinement of another binary relation R_2, written $R_1 \prec R_2$, if

$$\forall s_1, s_2 \in S.\ s_1 R_1 s_2 \ \rightarrow\ s_1 R_2 s_2\ .$$

- Example 1: Consider set $S = \{a, b\}$, and relations $R_1 = \{\langle a, b\rangle\}$ and $R_2 = \{\langle a, b\rangle, \langle b, b\rangle\}$
- Do either of these hold? $R_1 \prec R_2$ or $R_2 \prec R_1$? $R_1 \prec R_2$
- Example 2: Consider set $\mathbb{Z}$ and the relations:
  $R_1 : \{xR_1y\ :\ x \bmod 2 = y \bmod 2\}$ $\quad R_2 : \{xR_2y\ :\ x \bmod 4 = y \bmod 4\}$
- What is the refinement relationship between R_1 and R_2?

Relation Refinements

- A binary relation R_1 is a refinement of another binary relation R_2, written $R_1 \prec R_2$, if

$$\forall s_1, s_2 \in S.\ s_1 R_1 s_2 \ \rightarrow \ s_1 R_2 s_2 \ .$$

- Example 1: Consider set $S = \{a, b\}$, and relations $R_1 = \{\langle a, b\rangle\}$ and $R_2 = \{\langle a, b\rangle, \langle b, b\rangle\}$
- Do either of these hold? $R_1 \prec R_2$ or $R_2 \prec R_1$? $R_1 \prec R_2$
- Example 2: Consider set $\mathbb{Z}$ and the relations:
  $R_1 : \{xR_1 y \ : \ x \text{ mod } 2 = y \text{ mod } 2\}$ $\quad R_2 : \{xR_2 y \ : \ x \text{ mod } 4 = y \text{ mod } 4\}$
- What is the refinement relationship between R_1 and R_2? $R_2 \prec R_1$

Equivalence Closure

- The equivalence closure R^E of a binary relation R over S is the equivalence relation such that:

Equivalence Closure

- The equivalence closure R^E of a binary relation R over S is the equivalence relation such that:

  1. R refines R^E, i.e. $R \prec R^E$;

Equivalence Closure

- The equivalence closure R^E of a binary relation R over S is the equivalence relation such that:

  1. R refines R^E, i.e. $R \prec R^E$;
  2. for all other equivalence relations R' s.t. $R \prec R'$, either $R' = R^E$ or $R^E \prec R'$

Equivalence Closure

- The equivalence closure R^E of a binary relation R over S is the equivalence relation such that:

  1. R refines R^E, i.e. $R \prec R^E$;
  2. for all other equivalence relations R' s.t. $R \prec R'$, either $R' = R^E$ or $R^E \prec R'$

- Thus, R^E is the smallest equivalence relation that includes R.

Equivalence Closure Example

- Consider set $S = \{a, b, c, d\}$ and binary relation

$$R : \{\langle a, b\rangle, \langle b, c\rangle, \langle d, d\rangle\}$$

Equivalence Closure Example

- Consider set $S = \{a, b, c, d\}$ and binary relation

$$R : \{\langle a, b\rangle, \langle b, c\rangle, \langle d, d\rangle\}$$

- Is R an equivalence relation?

Equivalence Closure Example

- Consider set $S = \{a, b, c, d\}$ and binary relation

$$R : \{\langle a, b\rangle, \langle b, c\rangle, \langle d, d\rangle\}$$

- Is R an equivalence relation? No

Equivalence Closure Example

- Consider set $S = \{a, b, c, d\}$ and binary relation

$$R : \{\langle a, b\rangle, \langle b, c\rangle, \langle d, d\rangle\}$$

- Is R an equivalence relation? No
- We want to compute the equivalence closure R^E, i.e., smallest equivalence relation including R

Equivalence Closure Example

- Consider set $S = \{a, b, c, d\}$ and binary relation

$$R : \{\langle a, b\rangle, \langle b, c\rangle, \langle d, d\rangle\}$$

- Is R an equivalence relation? No

- We want to compute the equivalence closure R^E, i.e., smallest equivalence relation including R

- Thus, R^E needs to include all tuples in R and must obey reflexivity, symmetry, and transitivity.

Equivalence Closure Example, cont

- $R : \{\langle a, b\rangle, \langle b, c\rangle, \langle d, d\rangle\}$

Equivalence Closure Example, cont

- $R : \{\langle a, b\rangle, \langle b, c\rangle, \langle d, d\rangle\}$
- Since R^E must include R, which elements are in R^E?

Equivalence Closure Example, cont

- $R : \{\langle a, b\rangle, \langle b, c\rangle, \langle d, d\rangle\}$
- Since R^E must include R, which elements are in R^E? $\langle a, b\rangle, \langle b, c\rangle, \langle d, d\rangle$

Equivalence Closure Example, cont

- $R : \{\langle a, b\rangle, \langle b, c\rangle, \langle d, d\rangle\}$
- Since R^E must include R, which elements are in R^E? $\langle a, b\rangle, \langle b, c\rangle, \langle d, d\rangle$
- Since R^E equivalence relation, it must obey reflexivity. What other elements in R^E due to reflexivity?

Equivalence Closure Example, cont

- $R : \{\langle a, b\rangle, \langle b, c\rangle, \langle d, d\rangle\}$
- Since R^E must include R, which elements are in R^E? $\langle a, b\rangle, \langle b, c\rangle, \langle d, d\rangle$
- Since R^E equivalence relation, it must obey reflexivity. What other elements in R^E due to reflexivity? $\langle a, a\rangle, \langle b, b\rangle, \langle c, c\rangle$

Equivalence Closure Example, cont

- $R : \{\langle a, b\rangle, \langle b, c\rangle, \langle d, d\rangle\}$
- Since R^E must include R, which elements are in R^E? $\langle a, b\rangle, \langle b, c\rangle, \langle d, d\rangle$
- Since R^E equivalence relation, it must obey reflexivity. What other elements in R^E due to reflexivity? $\langle a, a\rangle, \langle b, b\rangle, \langle c, c\rangle$
- What elements in R^E due to symmetry?

Equivalence Closure Example, cont

- $R : \{\langle a, b\rangle, \langle b, c\rangle, \langle d, d\rangle\}$
- Since R^E must include R, which elements are in R^E? $\langle a, b\rangle, \langle b, c\rangle, \langle d, d\rangle$
- Since R^E equivalence relation, it must obey reflexivity. What other elements in R^E due to reflexivity? $\langle a, a\rangle, \langle b, b\rangle, \langle c, c\rangle$
- What elements in R^E due to symmetry? $\langle b, a\rangle, \langle c, b\rangle$

Equivalence Closure Example, cont

- $R : \{\langle a, b\rangle, \langle b, c\rangle, \langle d, d\rangle\}$
- Since R^E must include R, which elements are in R^E? $\langle a, b\rangle, \langle b, c\rangle, \langle d, d\rangle$
- Since R^E equivalence relation, it must obey reflexivity. What other elements in R^E due to reflexivity? $\langle a, a\rangle, \langle b, b\rangle, \langle c, c\rangle$
- What elements in R^E due to symmetry? $\langle b, a\rangle, \langle c, b\rangle$
- What elements in R^E due to transitivity?

Equivalence Closure Example, cont

- $R : \{\langle a, b\rangle, \langle b, c\rangle, \langle d, d\rangle\}$
- Since R^E must include R, which elements are in R^E? $\langle a, b\rangle, \langle b, c\rangle, \langle d, d\rangle$
- Since R^E equivalence relation, it must obey reflexivity. What other elements in R^E due to reflexivity? $\langle a, a\rangle, \langle b, b\rangle, \langle c, c\rangle$
- What elements in R^E due to symmetry? $\langle b, a\rangle, \langle c, b\rangle$
- What elements in R^E due to transitivity? $\langle a, c\rangle, \langle c, a\rangle$

Equivalence Closure Example, cont

- $R : \{\langle a, b\rangle, \langle b, c\rangle, \langle d, d\rangle\}$
- Since R^E must include R, which elements are in R^E? $\langle a, b\rangle, \langle b, c\rangle, \langle d, d\rangle$
- Since R^E equivalence relation, it must obey reflexivity. What other elements in R^E due to reflexivity? $\langle a, a\rangle, \langle b, b\rangle, \langle c, c\rangle$
- What elements in R^E due to symmetry? $\langle b, a\rangle, \langle c, b\rangle$
- What elements in R^E due to transitivity? $\langle a, c\rangle, \langle c, a\rangle$
- What is R^E?

Equivalence Closure Example, cont

- $R : \{\langle a, b\rangle, \langle b, c\rangle, \langle d, d\rangle\}$
- Since R^E must include R, which elements are in R^E? $\langle a, b\rangle, \langle b, c\rangle, \langle d, d\rangle$
- Since R^E equivalence relation, it must obey reflexivity. What other elements in R^E due to reflexivity? $\langle a, a\rangle, \langle b, b\rangle, \langle c, c\rangle$
- What elements in R^E due to symmetry? $\langle b, a\rangle, \langle c, b\rangle$
- What elements in R^E due to transitivity? $\langle a, c\rangle, \langle c, a\rangle$
- What is R^E?

  $R^E = \{\langle a, b\rangle, \langle b, c\rangle, \langle d, d\rangle, \langle a, b\rangle, \langle b, c\rangle, \langle d, d\rangle, \langle b, a\rangle, \langle c, b\rangle, \langle a, c\rangle, \langle c, a\rangle\}$

Equivalence Closure Example, cont

- $R : \{\langle a, b\rangle, \langle b, c\rangle, \langle d, d\rangle\}$
- $R^E = \{\langle a, b\rangle, \langle b, c\rangle, \langle d, d\rangle, \langle a, b\rangle, \langle b, c\rangle, \langle d, d\rangle, \langle b, a\rangle, \langle c, b\rangle, \langle a, c\rangle, \langle c, a\rangle\}$

Equivalence Closure Example, cont

- $R : \{\langle a, b\rangle, \langle b, c\rangle, \langle d, d\rangle\}$
- $R^E = \{\langle a, b\rangle, \langle b, c\rangle, \langle d, d\rangle, \langle a, b\rangle, \langle b, c\rangle, \langle d, d\rangle, \langle b, a\rangle, \langle c, b\rangle, \langle a, c\rangle, \langle c, a\rangle\}$
- Consider relation $R' = R^E \cup \{\langle c, d\rangle, \langle d, c\rangle, \langle b, d\rangle, \langle d, b\rangle, \langle a, d\rangle, \langle d, a\rangle\}$

Equivalence Closure Example, cont

- $R : \{\langle a, b\rangle, \langle b, c\rangle, \langle d, d\rangle\}$
- $R^E = \{\langle a, b\rangle, \langle b, c\rangle, \langle d, d\rangle, \langle a, b\rangle, \langle b, c\rangle, \langle d, d\rangle, \langle b, a\rangle, \langle c, b\rangle, \langle a, c\rangle, \langle c, a\rangle\}$
- Consider relation $R' = R^E \cup \{\langle c, d\rangle, \langle d, c\rangle, \langle b, d\rangle, \langle d, b\rangle, \langle a, d\rangle, \langle d, a\rangle\}$
- R' is also an equivalence relation and covers R

Equivalence Closure Example, cont

- $R : \{\langle a, b\rangle, \langle b, c\rangle, \langle d, d\rangle\}$
- $R^E = \{\langle a, b\rangle, \langle b, c\rangle, \langle d, d\rangle, \langle a, b\rangle, \langle b, c\rangle, \langle d, d\rangle, \langle b, a\rangle, \langle c, b\rangle, \langle a, c\rangle, \langle c, a\rangle\}$
- Consider relation $R' = R^E \cup \{\langle c, d\rangle, \langle d, c\rangle, \langle b, d\rangle, \langle d, b\rangle, \langle a, d\rangle, \langle d, a\rangle\}$
- R' is also an equivalence relation and covers R
- Is R' also an equivalence closure of R?

Equivalence Closure Example, cont

- $R : \{\langle a, b\rangle, \langle b, c\rangle, \langle d, d\rangle\}$
- $R^E = \{\langle a, b\rangle, \langle b, c\rangle, \langle d, d\rangle, \langle a, b\rangle, \langle b, c\rangle, \langle d, d\rangle, \langle b, a\rangle, \langle c, b\rangle, \langle a, c\rangle, \langle c, a\rangle\}$
- Consider relation $R' = R^E \cup \{\langle c, d\rangle, \langle d, c\rangle, \langle b, d\rangle, \langle d, b\rangle, \langle a, d\rangle, \langle d, a\rangle\}$
- R' is also an equivalence relation and covers R
- Is R' also an equivalence closure of R? No!

Congruence Closure

- Given a set S and binary relation R, we also define congruence closure of R

Congruence Closure

- Given a set S and binary relation R, we also define congruence closure of R
- Congruence closure is similar to equivalence closure, but it is the smallest congruence relation that covers R

Congruence Closure

- Given a set S and binary relation R, we also define congruence closure of R
- Congruence closure is similar to equivalence closure, but it is the smallest congruence relation that covers R
- Formally, the congruence closure R^C of a binary relation R over S is the congruence relation such that:

  1. R refines R^C, i.e. $R \prec R^C$;
  2. for all other congruence relations R' s.t. $R \prec R'$, either $R' = R^C$ or $R^C \prec R'$

Congruence Closure Algorithm

- The decision procedure for $T_=$ computes congruence closure of equality over the subterm set of formula

Congruence Closure Algorithm

- The decision procedure for $T_=$ computes congruence closure of equality over the subterm set of formula
- Subterm set S_F of F is the set of all subterms of F

Congruence Closure Algorithm

- The decision procedure for $T_=$ computes congruence closure of equality over the subterm set of formula
- Subterm set S_F of F is the set of all subterms of F
- Example: Consider formula $F : f(a, b) = a \land f(f(a, b), b) \neq a$

Congruence Closure Algorithm

- The decision procedure for $T_=$ computes congruence closure of equality over the subterm set of formula
- Subterm set S_F of F is the set of all subterms of F
- Example: Consider formula $F : f(a, b) = a \land f(f(a, b), b) \neq a$
- What is S_F?

Congruence Closure Algorithm

- The decision procedure for $T_=$ computes congruence closure of equality over the subterm set of formula
- Subterm set S_F of F is the set of all subterms of F
- Example: Consider formula $F : f(a,b) = a \land f(f(a,b),b) \neq a$
- What is S_F? $\{a, b, f(a,b), f(f(a,b),b)\}$

Satisfiability using Congruence Relations

- We can now define satisfiability of a $\Sigma_=$ formula in terms of congruence closure over subterm set

Satisfiability using Congruence Relations

- We can now define satisfiability of a $\Sigma_=$ formula in terms of congruence closure over subterm set
- Consider $\Sigma_=$ formula F:

$$F: \; s_1 = t_1 \wedge \ldots s_m = t_m \wedge s_{m+1} \neq t_{m+1} \wedge \ldots s_n \neq t_n$$

Satisfiability using Congruence Relations

- We can now define satisfiability of a $\Sigma_=$ formula in terms of congruence closure over subterm set
- Consider $\Sigma_=$ formula F:

$$F: \; s_1 = t_1 \wedge \ldots s_m = t_m \wedge s_{m+1} \neq t_{m+1} \wedge \ldots s_n \neq t_n$$

- Theorem: F is satisfiable iff there exists a congruence relation $\sim$ over the subterm set S_F of F such that:

Satisfiability using Congruence Relations

- We can now define satisfiability of a $\Sigma_=$ formula in terms of congruence closure over subterm set
- Consider $\Sigma_=$ formula F:

$$F: \ s_1 = t_1 \land \ldots s_m = t_m \land s_{m+1} \neq t_{m+1} \land \ldots s_n \neq t_n$$

- Theorem: F is satisfiable iff there exists a congruence relation $\sim$ over the subterm set S_F of F such that:
  1. For each i in $[1, m]$, $s_i \sim t_i$

Satisfiability using Congruence Relations

- We can now define satisfiability of a $\Sigma_=$ formula in terms of congruence closure over subterm set
- Consider $\Sigma_=$ formula F:

$$F: \; s_1 = t_1 \wedge \dots s_m = t_m \wedge s_{m+1} \neq t_{m+1} \wedge \dots s_n \neq t_n$$

- Theorem: F is satisfiable iff there exists a congruence relation $\sim$ over the subterm set S_F of F such that:
  1. For each i in $[1, m]$, $s_i \sim t_i$
  2. For each i in $[m+1, n]$, $s_i \not\sim t_i$

Congruence Closure Algorithm: Basic Idea

Congruence closure algorithm decide satisfiability of

$$F: \; s_1 = t_1 \wedge \ldots s_m = t_m \wedge s_{m+1} \neq t_{m+1} \wedge \ldots s_n \neq t_n$$

Congruence Closure Algorithm: Basic Idea

Congruence closure algorithm decide satisfiability of

$$F: \; s_1 = t_1 \wedge \ldots s_m = t_m \wedge s_{m+1} \neq t_{m+1} \wedge \ldots s_n \neq t_n$$

1. Construct the congruence closure $\sim$ of

$$\{s_1 = t_1, \ldots, s_m = t_m\}$$

   over the subterm set S_F.

Congruence Closure Algorithm: Basic Idea

Congruence closure algorithm decide satisfiability of

$$F: \; s_1 = t_1 \wedge \ldots s_m = t_m \wedge s_{m+1} \neq t_{m+1} \wedge \ldots s_n \neq t_n$$

1. Construct the congruence closure $\sim$ of

   $$\{s_1 = t_1, \ldots, s_m = t_m\}$$

   over the subterm set S_F.

2. If $s_i \sim t_i$ for any i in $[m+1, n]$, F is unsatisfiable

Congruence Closure Algorithm: Basic Idea

Congruence closure algorithm decide satisfiability of

$$F: \; s_1 = t_1 \wedge \ldots s_m = t_m \wedge s_{m+1} \neq t_{m+1} \wedge \ldots s_n \neq t_n$$

1. Construct the congruence closure $\sim$ of

$$\{s_1 = t_1, \ldots, s_m = t_m\}$$

   over the subterm set S_F.

2. If $s_i \sim t_i$ for any i in $[m+1, n]$, F is unsatisfiable

3. Otherwise, F is satisfiable

Example

- Consider the formula $F : f(a, b) = a \land f(f(a, b), b) \neq a$

Example

- Consider the formula $F : f(a,b) = a \wedge f(f(a,b),b) \neq a$
- We'll represent $\sim$ as a set of congruence classes, i.e., if t_1 and t_2 are in the same set, this means $t_1 \sim t_2$, otherwise $t_1 \not\sim t_2$

Example

- Consider the formula $F : f(a, b) = a \wedge f(f(a, b), b) \neq a$
- We'll represent $\sim$ as a set of congruence classes, i.e., if t_1 and t_2 are in the same set, this means $t_1 \sim t_2$, otherwise $t_1 \not\sim t_2$
- First, construct subterm set S_F and place each subterm in a separate set:

Example

- Consider the formula $F : f(a, b) = a \wedge f(f(a, b), b) \neq a$
- We'll represent $\sim$ as a set of congruence classes, i.e., if t_1 and t_2 are in the same set, this means $t_1 \sim t_2$, otherwise $t_1 \not\sim t_2$
- First, construct subterm set S_F and place each subterm in a separate set:

$$\{\{a\}, \{b\}, \{f(a, b)\}, \{f(f(a, b), b)\}\}$$

Example

- Consider the formula $F : f(a, b) = a \wedge f(f(a, b), b) \neq a$
- We'll represent $\sim$ as a set of congruence classes, i.e., if t_1 and t_2 are in the same set, this means $t_1 \sim t_2$, otherwise $t_1 \not\sim t_2$
- First, construct subterm set S_F and place each subterm in a separate set:

$$\{\{a\}, \{b\}, \{f(a, b)\}, \{f(f(a, b), b)\}\}$$

- Because of equality $f(a, b) = a$, merge congruence classes of $f(a, b)$ and a:

Example

- Consider the formula $F : f(a,b) = a \wedge f(f(a,b),b) \neq a$
- We'll represent $\sim$ as a set of congruence classes, i.e., if t_1 and t_2 are in the same set, this means $t_1 \sim t_2$, otherwise $t_1 \not\sim t_2$
- First, construct subterm set S_F and place each subterm in a separate set:

$$\{\{a\},\{b\},\{f(a,b)\},\{f(f(a,b),b)\}\}$$

- Because of equality $f(a,b) = a$, merge congruence classes of $f(a,b)$ and a:

$$\{\{a, f(a,b)\},\{b\},\{f(f(a,b),b)\}\}$$

Example, cont

- Formula $F : f(a, b) = a \wedge f(f(a, b), b) \neq a$
- Current congruence classes:

$$\{\{a, f(a, b)\}, \{b\}, \{f(f(a, b), b)\}\}$$

Example, cont

- Formula $F : f(a, b) = a \land f(f(a, b), b) \neq a$
- Current congruence classes:

$$\{\{a, f(a, b)\}, \{b\}, \{f(f(a, b), b)\}\}$$

- Using $a \sim f(a, b)$ and $b \sim b$, what does function congruence imply?

Example, cont

- Formula $F : f(a, b) = a \land f(f(a, b), b) \neq a$
- Current congruence classes:

$$\{\{a, f(a, b)\}, \{b\}, \{f(f(a, b), b)\}\}$$

- Using $a \sim f(a, b)$ and $b \sim b$, what does function congruence imply?
  $f(f(a, b), b) = f(a, b)$

Example, cont

- Formula $F : f(a, b) = a \wedge f(f(a, b), b) \neq a$
- Current congruence classes:

$$\{\{a, f(a, b)\}, \{b\}, \{f(f(a, b), b)\}\}$$

- Using $a \sim f(a, b)$ and $b \sim b$, what does function congruence imply?
  $f(f(a, b), b) = f(a, b)$
- Thus, merge congruence classes of $f(a, b)$ and $f(f(a, b), b)$:

$$\{\{a, f(a, b), f(f(a, b), b)\}, \{b\}\}$$

Example, cont

- Formula $F : f(a, b) = a \wedge f(f(a, b), b) \neq a$
- Current congruence classes:

$$\{\{a, f(a, b)\}, \{b\}, \{f(f(a, b), b)\}\}$$

- Using $a \sim f(a, b)$ and $b \sim b$, what does function congruence imply? $f(f(a, b), b) = f(a, b)$
- Thus, merge congruence classes of $f(a, b)$ and $f(f(a, b), b)$:

$$\{\{a, f(a, b), f(f(a, b), b)\}, \{b\}\}$$

- This represents the congruence closure over S_F.

Example, cont

- Formula $F : f(a, b) = a \wedge f(f(a, b), b) \neq a$
- Congruence closure: $\{\{a, f(a, b), f(f(a, b), b)\}, \{b\}\}$
- Is F satisfiable?

Example, cont

- Formula $F : f(a, b) = a \wedge f(f(a, b), b) \neq a$
- Congruence closure: $\{\{a, f(a, b), f(f(a, b), b)\}, \{b\}\}$
- Is F satisfiable? No
- Since a and $f(f(a, b), b)$ are in same congruence class, we have $a \sim f(f(a, b), b)$
- This contradicts $f(f(a, b), b) \neq a$!

Another Example

- Consider formula:

$$F : f(f(f(a))) = a \land f(f(f(f(f(a))))) = a \land f(a) \neq a$$

Another Example

- Consider formula:

$$F : f(f(f(a))) = a \wedge f(f(f(f(f(a))))) = a \wedge f(a) \neq a$$

- What is the subterm set S_F?

Another Example

- Consider formula:

$$F : f(f(f(a))) = a \land f(f(f(f(f(a))))) = a \land f(a) \neq a$$

- What is the subterm set S_F?

$$\{a, f(a), f^2(a), f^3(a), f^4(a), f^5(a)\}$$

Another Example

- Consider formula:

$$F : f(f(f(a))) = a \land f(f(f(f(f(a))))) = a \land f(a) \neq a$$

- What is the subterm set S_F?

$$\{a, f(a), f^2(a), f^3(a), f^4(a), f^5(a)\}$$

- Initially, place each subterm in its own congruence class:

$$\{\{a\}, \{f(a)\}, \{f^2(a)\}, \{f^3(a)\}, \{f^4(a)\}, \{f^5(a)\}\}$$

Another Example

- Consider formula:

$$F : f(f(f(a))) = a \wedge f(f(f(f(f(a))))) = a \wedge f(a) \neq a$$

- What is the subterm set S_F?

$$\{a, f(a), f^2(a), f^3(a), f^4(a), f^5(a)\}$$

- Initially, place each subterm in its own congruence class:

$$\{\{a\}, \{f(a)\}, \{f^2(a)\}, \{f^3(a)\}, \{f^4(a)\}, \{f^5(a)\}\}$$

- Because of equality $f^3(a) = a$, $f^3(a)$ and a are placed in same congruence class:

Another Example

- Consider formula:

$$F : f(f(f(a))) = a \wedge f(f(f(f(f(a))))) = a \wedge f(a) \neq a$$

- What is the subterm set S_F?

$$\{a, f(a), f^2(a), f^3(a), f^4(a), f^5(a)\}$$

- Initially, place each subterm in its own congruence class:

$$\{\{a\}, \{f(a)\}, \{f^2(a)\}, \{f^3(a)\}, \{f^4(a)\}, \{f^5(a)\}\}$$

- Because of equality $f^3(a) = a$, $f^3(a)$ and a are placed in same congruence class:

$$\{\{a, f^3(a)\}, \{f(a)\}, \{f^2(a)\}, \{f^4(a)\}, \{f^5(a)\}\}$$

Another Example, cont

- Formula $F : f^3(a) = a \land f^5(a) = a \land f(a) \neq a$
- Current congruence classes:

$$\{\{a, f^3(a)\}, \{f(a)\}, \{f^2(a)\}, \{f^4(a)\}, \{f^5(a)\}\}$$

Another Example, cont

- Formula $F : f^3(a) = a \wedge f^5(a) = a \wedge f(a) \neq a$
- Current congruence classes:

$$\{\{a, f^3(a)\}, \{f(a)\}, \{f^2(a)\}, \{f^4(a)\}, \{f^5(a)\}\}$$

- From $a = f^3(a)$, what can we infer using function congruence?

Another Example, cont

- Formula $F : f^3(a) = a \wedge f^5(a) = a \wedge f(a) \neq a$
- Current congruence classes:

$$\{\{a, f^3(a)\}, \{f(a)\}, \{f^2(a)\}, \{f^4(a)\}, \{f^5(a)\}\}$$

- From $a = f^3(a)$, what can we infer using function congruence?
  $f(a) = f^4(a)$ and $f^2(a) = f^5(a)$

Another Example, cont

- Formula $F : f^3(a) = a \wedge f^5(a) = a \wedge f(a) \neq a$
- Current congruence classes:

$$\{\{a, f^3(a)\}, \{f(a)\}, \{f^2(a)\}, \{f^4(a)\}, \{f^5(a)\}\}$$

- From $a = f^3(a)$, what can we infer using function congruence?
  $f(a) = f^4(a)$ and $f^2(a) = f^5(a)$
- Resulting congruence classes:

$$\{\{a, f^3(a)\}, \{f(a), f^4(a)\}, \{f^2(a), f^5(a)\}\}$$

Another Example, cont

- Formula $F : f^3(a) = a \wedge f^5(a) = a \wedge f(a) \neq a$
- Current congruence classes:

$$\{\{a, f^3(a)\}, \{f(a), f^4(a)\}, \{f^2(a), f^5(a)\}\}$$

Another Example, cont

- Formula $F : f^3(a) = a \wedge f^5(a) = a \wedge f(a) \neq a$
- Current congruence classes:

$$\{\{a, f^3(a)\}, \{f(a), f^4(a)\}, \{f^2(a), f^5(a)\}\}$$

- Now, process equality $f^5(a) = a$; which classes do we merge?

Another Example, cont

- Formula $F : f^3(a) = a \wedge f^5(a) = a \wedge f(a) \neq a$
- Current congruence classes:

$$\{\{a, f^3(a)\}, \{f(a), f^4(a)\}, \{f^2(a), f^5(a)\}\}$$

- Now, process equality $f^5(a) = a$; which classes do we merge?

$$\{\{a, f^3(a), f^2(a), f^5(a)\}, \{f(a), f^4(a)\}\}$$

Another Example, cont

- Formula $F : f^3(a) = a \wedge f^5(a) = a \wedge f(a) \neq a$
- Current congruence classes:

$$\{\{a, f^3(a)\}, \{f(a), f^4(a)\}, \{f^2(a), f^5(a)\}\}$$

- Now, process equality $f^5(a) = a$; which classes do we merge?

$$\{\{a, f^3(a), f^2(a), f^5(a)\}, \{f(a), f^4(a)\}\}$$

- From $a = f^2(a)$, what can we infer via function congruence?

Another Example, cont

- Formula $F : f^3(a) = a \wedge f^5(a) = a \wedge f(a) \neq a$
- Current congruence classes:

$$\{\{a, f^3(a)\}, \{f(a), f^4(a)\}, \{f^2(a), f^5(a)\}\}$$

- Now, process equality $f^5(a) = a$; which classes do we merge?

$$\{\{a, f^3(a), f^2(a), f^5(a)\}, \{f(a), f^4(a)\}\}$$

- From $a = f^2(a)$, what can we infer via function congruence? $f(a) = f^3(a)$

Another Example, cont

- Formula $F : f^3(a) = a \wedge f^5(a) = a \wedge f(a) \neq a$
- Current congruence classes:

$$\{\{a, f^3(a)\}, \{f(a), f^4(a)\}, \{f^2(a), f^5(a)\}\}$$

- Now, process equality $f^5(a) = a$; which classes do we merge?

$$\{\{a, f^3(a), f^2(a), f^5(a)\}, \{f(a), f^4(a)\}\}$$

- From $a = f^2(a)$, what can we infer via function congruence? $f(a) = f^3(a)$
- Thus, merge the two congruence classes:

$$\{\{a, f(a), f^2(a), f^3(a), f^4(a), f^5(a)\}\}$$

Another Example, cont

- Formula $F : f^3(a) = a \land f^5(a) = a \land f(a) \neq a$
- Currenct congruence classes:

$$\{\{a, f(a), f^2(a), f^3(a), f^4(a), f^5(a)\}\}$$

Another Example, cont

- Formula $F : f^3(a) = a \land f^5(a) = a \land f(a) \neq a$
- Currenct congruence classes:

$$\{\{a, f(a), f^2(a), f^3(a), f^4(a), f^5(a)\}\}$$

- Is the formula satisfiable?

Another Example, cont

- Formula $F : f^3(a) = a \wedge f^5(a) = a \wedge f(a) \neq a$
- Currenct congruence classes:

$$\{\{a, f(a), f^2(a), f^3(a), f^4(a), f^5(a)\}\}$$

- Is the formula satisfiable? No
- Since $f(a)$ and a are in same congruence class, this contradicts $f(a) \neq a$

One More Example

- Consider formula $F : f(x) = f(y) \land x \neq y$

One More Example

- Consider formula $F : f(x) = f(y) \wedge x \neq y$
- What is the subterm set?

One More Example

- Consider formula $F : f(x) = f(y) \wedge x \neq y$
- What is the subterm set? $\{x, y, f(x), f(y)\}$

One More Example

- Consider formula $F : f(x) = f(y) \wedge x \neq y$
- What is the subterm set? $\{x, y, f(x), f(y)\}$
- Each subterm starts in its own congruence class: $\{\{x\}, \{y\}, \{f(x)\}, \{f(y)\}\}$

One More Example

- Consider formula $F : f(x) = f(y) \wedge x \neq y$
- What is the subterm set? $\{x, y, f(x), f(y)\}$
- Each subterm starts in its own congruence class: $\{\{x\}, \{y\}, \{f(x)\}, \{f(y)\}\}$
- Process equality $f(x) = f(y) \Rightarrow$

One More Example

- Consider formula $F : f(x) = f(y) \wedge x \neq y$
- What is the subterm set? $\{x, y, f(x), f(y)\}$
- Each subterm starts in its own congruence class: $\{\{x\}, \{y\}, \{f(x)\}, \{f(y)\}\}$
- Process equality $f(x) = f(y) \Rightarrow \{\{x\}, \{y\}, \{f(x), f(y)\}\}$

One More Example

- Consider formula $F : f(x) = f(y) \land x \neq y$
- What is the subterm set? $\{x, y, f(x), f(y)\}$
- Each subterm starts in its own congruence class: $\{\{x\}, \{y\}, \{f(x)\}, \{f(y)\}\}$
- Process equality $f(x) = f(y) \Rightarrow \{\{x\}, \{y\}, \{f(x), f(y)\}\}$
- What new equalities can we infer from congruence?

One More Example

- Consider formula $F : f(x) = f(y) \wedge x \neq y$
- What is the subterm set? $\{x, y, f(x), f(y)\}$
- Each subterm starts in its own congruence class: $\{\{x\}, \{y\}, \{f(x)\}, \{f(y)\}\}$
- Process equality $f(x) = f(y) \Rightarrow \{\{x\}, \{y\}, \{f(x), f(y)\}\}$
- What new equalities can we infer from congruence? None!

One More Example

- Consider formula $F : f(x) = f(y) \wedge x \neq y$
- What is the subterm set? $\{x, y, f(x), f(y)\}$
- Each subterm starts in its own congruence class: $\{\{x\}, \{y\}, \{f(x)\}, \{f(y)\}\}$
- Process equality $f(x) = f(y) \Rightarrow \{\{x\}, \{y\}, \{f(x), f(y)\}\}$
- What new equalities can we infer from congruence? None!
- Is the formula satisfiable?

One More Example

- Consider formula $F : f(x) = f(y) \land x \neq y$
- What is the subterm set? $\{x, y, f(x), f(y)\}$
- Each subterm starts in its own congruence class:
  $\{\{x\}, \{y\}, \{f(x)\}, \{f(y)\}\}$
- Process equality $f(x) = f(y) \Rightarrow \{\{x\}, \{y\}, \{f(x), f(y)\}\}$
- What new equalities can we infer from congruence? None!
- Is the formula satisfiable? Yes

How to Compute Congruence Closure

- So far, we described how to decide satisfiability using congruence closure

How to Compute Congruence Closure

- So far, we described how to decide satisfiability using congruence closure
- But we haven't discussed an algorithm for efficiently computing congruence closure

How to Compute Congruence Closure

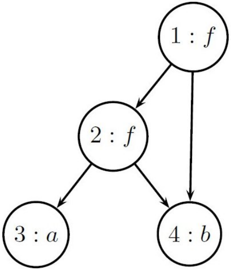
- So far, we described how to decide satisfiability using congruence closure
- But we haven't discussed an algorithm for efficiently computing congruence closure
- There is a very efficient algorithm called Union-Find for computing congruence classes

How to Compute Congruence Closure

- So far, we described how to decide satisfiability using congruence closure
- But we haven't discussed an algorithm for efficiently computing congruence closure
- There is a very efficient algorithm called Union-Find for computing congruence classes
- Next: Talk about Union-Find algorithm for computing congruence closures

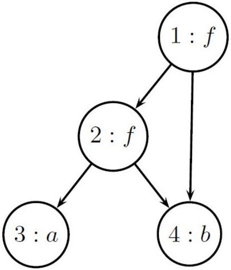
Representing Subterms

- To compute congruence closure efficiently, we'll represent the subterm set of the formula as a DAG



Representing Subterms

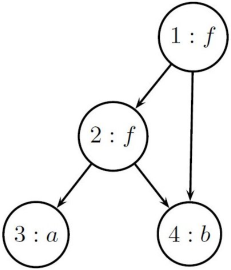
- To compute congruence closure efficiently, we'll represent the subterm set of the formula as a DAG



- Each node corresponds to a subterm and has unique id

Representing Subterms

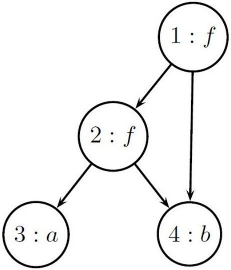
- To compute congruence closure efficiently, we'll represent the subterm set of the formula as a DAG



- Each node corresponds to a subterm and has unique id
- Edges point from function symbol to arguments

Representing Subterms

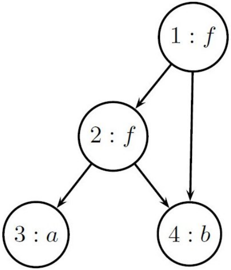
- To compute congruence closure efficiently, we'll represent the subterm set of the formula as a DAG



- Each node corresponds to a subterm and has unique id
- Edges point from function symbol to arguments
- Question: What subterm does node labeled 1 represent?

Representing Subterms

- To compute congruence closure efficiently, we'll represent the subterm set of the formula as a DAG



- Each node corresponds to a subterm and has unique id
- Edges point from function symbol to arguments
- Question: What subterm does node labeled 1 represent? f(f(a,b), b)

Representative of Congruence Class

- To compute congruence closure, we need to merge congruence classes

Representative of Congruence Class

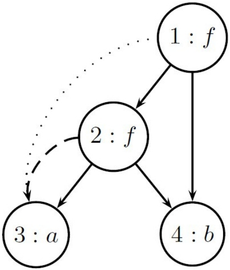
- To compute congruence closure, we need to merge congruence classes
- To do this efficiently, each congruence class has a representative: When merging two classes, only need to update the representative

Representative of Congruence Class

- To compute congruence closure, we need to merge congruence classes
- To do this efficiently, each congruence class has a representative: When merging two classes, only need to update the representative
  - Thus, for a given subterm, we need to be able to find the representative of its class

Representative of Congruence Class

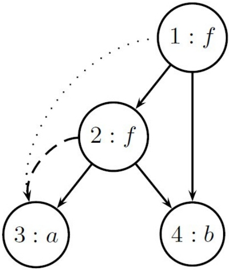
- To compute congruence closure, we need to merge congruence classes
- To do this efficiently, each congruence class has a representative: When merging two classes, only need to update the representative



- Thus, for a given subterm, we need to be able to find the representative of its class
- Each subterm contains a pointer that eventually leads to the representative of its congruence class

Representative of Congruence Class

- To compute congruence closure, we need to merge congruence classes
- To do this efficiently, each congruence class has a representative: When merging two classes, only need to update the representative



- Thus, for a given subterm, we need to be able to find the representative of its class
- Each subterm contains a pointer that eventually leads to the representative of its congruence class
- In this example, $a, f(a, b), f(f(a, b), b)$ are in same congruence class; a is the representative

Parents of a Subterm

- In addition to efficiently finding representative, also need to efficiently find parents of terms

Parents of a Subterm

- In addition to efficiently finding representative, also need to efficiently find parents of terms
- Why? Because if $x_1 = y_1, \ldots, x_k = y_k$, function congruence implies $f(\vec{x}) = f(\vec{y})$

Parents of a Subterm

- In addition to efficiently finding representative, also need to efficiently find parents of terms
- Why? Because if $x_1 = y_1, \ldots, x_k = y_k$, function congruence implies $f(\vec{x}) = f(\vec{y})$
- Thus, when each x_i, y_i pair is in same conguence class, need to merge congruence classes of their parents $f(\vec{x})$ and $f(\vec{y})$

Parents of a Subterm

- In addition to efficiently finding representative, also need to efficiently find parents of terms
- Why? Because if $x_1 = y_1, \ldots, x_k = y_k$, function congruence implies $f(\vec{x}) = f(\vec{y})$
- Thus, when each x_i, y_i pair is in same conguence class, need to merge congruence classes of their parents $f(\vec{x})$ and $f(\vec{y})$
- Thus, keep pointer from representative of congruence class to parents of all subterms in the congruence class

Summary of Data Structure

- Represent subterms as a DAG

Summary of Data Structure

- Represent subterms as a DAG
- Each node in the DAG corresponds to a subterm

Summary of Data Structure

- Represent subterms as a DAG
- Each node in the DAG corresponds to a subterm
- Each node stores its unique id, name of function or variable, and list of argument subterms

Summary of Data Structure

- Represent subterms as a DAG
- Each node in the DAG corresponds to a subterm
- Each node stores its unique id, name of function or variable, and list of argument subterms
- Each node n has a find pointer field that leads to its representative

Summary of Data Structure

- Represent subterms as a DAG
- Each node in the DAG corresponds to a subterm
- Each node stores its unique id, name of function or variable, and list of argument subterms
- Each node n has a find pointer field that leads to its representative
- The find field of a representative points to itself

Summary of Data Structure

- Represent subterms as a DAG
- Each node in the DAG corresponds to a subterm
- Each node stores its unique id, name of function or variable, and list of argument subterms
- Each node n has a find pointer field that leads to its representative
- The find field of a representative points to itself
- Each representative stores the set of parents for all subterms in that class

Summary of Data Structure

- Represent subterms as a DAG
- Each node in the DAG corresponds to a subterm
- Each node stores its unique id, name of function or variable, and list of argument subterms
- Each node n has a find pointer field that leads to its representative
- The find field of a representative points to itself
- Each representative stores the set of parents for all subterms in that class
- If a term is not a representative, then its parents field is empty

Finding Representative of Congruence Class

- Given a term t, we need to find representative for that term

Finding Representative of Congruence Class

- Given a term t, we need to find representative for that term
- If t's find field points to itself, then t is the representative of its congruence class

Finding Representative of Congruence Class

- Given a term t, we need to find representative for that term
- If t's find field points to itself, then t is the representative of its congruence class
- Otherwise, we follow the chain of find references until we find a node t' that points to itself

Finding Representative of Congruence Class

- Given a term t, we need to find representative for that term
- If t's find field points to itself, then t is the representative of its congruence class
- Otherwise, we follow the chain of find references until we find a node t' that points to itself
- In this case, t' is t's representative

Merging Congruence Classes

- Using this data structure, how do we merge congruence classes of two terms t_1 and t_2?

Merging Congruence Classes

- Using this data structure, how do we merge congruence classes of two terms t_1 and t_2?
- First find representatives of t_1 and t_2 as decribed earlier

Merging Congruence Classes

- Using this data structure, how do we merge congruence classes of two terms t_1 and t_2?
- First find representatives of t_1 and t_2 as decribed earlier
- Want to make $Rep(t_2)$ new representative for merged class

Merging Congruence Classes

- Using this data structure, how do we merge congruence classes of two terms t_1 and t_2?
- First find representatives of t_1 and t_2 as decribed earlier
- Want to make $Rep(t_2)$ new representative for merged class
- Thus, change find field of $Rep(t_1)$ to point to $Rep(t_2)$

Merging Congruence Classes

- Using this data structure, how do we merge congruence classes of two terms t_1 and t_2?
- First find representatives of t_1 and t_2 as decribed earlier
- Want to make $Rep(t_2)$ new representative for merged class
- Thus, change find field of $Rep(t_1)$ to point to $Rep(t_2)$
- Update parents: add parent terms stored in $Rep(t_1)$ to those of $Rep(t_2)$, and remove parents stored in $Rep(t_1)$

Processing Equalities

- How do we process an equality $t_1 = t_2$?

Processing Equalities

- How do we process an equality $t_1 = t_2$?
- Need to merge equivalence classes of t_1 and t_2

Processing Equalities

- How do we process an equality $t_1 = t_2$?
- Need to merge equivalence classes of t_1 and t_2
- Might potentially also need to merge t_1 and t_2's parents due to function congruence

Processing Equalities

- How do we process an equality $t_1 = t_2$?
- Need to merge equivalence classes of t_1 and t_2
- Might potentially also need to merge t_1 and t_2's parents due to function congruence
- Given parent p_1 of t_1 and p_2 of t_2, when do we merge p_1 and p_2's congruence classes?

Processing Equalities

- How do we process an equality $t_1 = t_2$?
- Need to merge equivalence classes of t_1 and t_2
- Might potentially also need to merge t_1 and t_2's parents due to function congruence
- Given parent p_1 of t_1 and p_2 of t_2, when do we merge p_1 and p_2's congruence classes?
- If they have the same function name and all of their arguments are congruent (i.e., have same representative)

Processing Equalities, cont

To process equality $t_1 = t_2$:

1. Find representatives of t_1 and t_2

Processing Equalities, cont

To process equality $t_1 = t_2$:

1. Find representatives of t_1 and t_2
2. Merge equivalence classes

Processing Equalities, cont

To process equality $t_1 = t_2$:

1. Find representatives of t_1 and t_2

2. Merge equivalence classes

3. Retrieve the set of parents P_1, P_2 stored in $Rep(t_1), Rep(t_2)$

Processing Equalities, cont

To process equality $t_1 = t_2$:

1. Find representatives of t_1 and t_2
2. Merge equivalence classes
3. Retrieve the set of parents P_1, P_2 stored in $Rep(t_1), Rep(t_2)$
4. For each $(p_i, p_j) \in P_1 \times P_2$, if p_i and p_j are congruent, process equality $p_i = p_j$

Processing Equalities, cont

To process equality $t_1 = t_2$:

1. Find representatives of t_1 and t_2
2. Merge equivalence classes
3. Retrieve the set of parents P_1, P_2 stored in $Rep(t_1), Rep(t_2)$
4. For each $(p_i, p_j) \in P_1 \times P_2$, if p_i and p_j are congruent, process equality $p_i = p_j$

Observe: Processing one equality creates new equalities, which in turn might generate other new equalities!

Question about Algorithm

- Recall: The representative stores parents of all terms in the congruence class

Question about Algorithm

- Recall: The representative stores parents of all terms in the congruence class
- Thus, when we process equality $t_1 = t_2$, we might also merge terms that are not t_1 and t_2's parents

Question about Algorithm

- Recall: The representative stores parents of all terms in the congruence class
- Thus, when we process equality $t_1 = t_2$, we might also merge terms that are not t_1 and t_2's parents
- Is this correct/necessary?

Question about Algorithm

- Recall: The representative stores parents of all terms in the congruence class
- Thus, when we process equality $t_1 = t_2$, we might also merge terms that are not t_1 and t_2's parents
- Is this correct/necessary? Yes!

Question about Algorithm

- Recall: The representative stores parents of all terms in the congruence class
- Thus, when we process equality $t_1 = t_2$, we might also merge terms that are not t_1 and t_2's parents
- Is this correct/necessary? Yes!
- If s_1, s_2 are in t_1, t_2's congruence class, $t_1 = t_2$ implies $s_1 = s_2$

Question about Algorithm

- Recall: The representative stores parents of all terms in the congruence class
- Thus, when we process equality $t_1 = t_2$, we might also merge terms that are not t_1 and t_2's parents
- Is this correct/necessary? Yes!
- If s_1, s_2 are in t_1, t_2's congruence class, $t_1 = t_2$ implies $s_1 = s_2$
- Thus, also need to process equality between s_1, s_2's parents

Question about Algorithm

- Recall: The representative stores parents of all terms in the congruence class
- Thus, when we process equality $t_1 = t_2$, we might also merge terms that are not t_1 and t_2's parents
- Is this correct/necessary? Yes!
- If s_1, s_2 are in t_1, t_2's congruence class, $t_1 = t_2$ implies $s_1 = s_2$
- Thus, also need to process equality between s_1, s_2's parents
- That's why representative stores all parents for cong. class

Full Algorithm for Deciding Satisfiability

Algorithm to decide satisfiability of $T_=$ formula

$$F: \ s_1 = t_1 \wedge \ldots s_m = t_m \wedge s_{m+1} \neq t_{m+1} \wedge \ldots s_n \neq t_n$$

Full Algorithm for Deciding Satisfiability

Algorithm to decide satisfiability of $T_=$ formula

$$F: \; s_1 = t_1 \wedge \ldots s_m = t_m \wedge s_{m+1} \neq t_{m+1} \wedge \ldots s_n \neq t_n$$

1. Compute subterms and construct initial DAG (each node's representative is itself)

Full Algorithm for Deciding Satisfiability

Algorithm to decide satisfiability of $T_=$ formula

$$F: \ s_1 = t_1 \wedge \ldots s_m = t_m \wedge s_{m+1} \neq t_{m+1} \wedge \ldots s_n \neq t_n$$

1. Compute subterms and construct initial DAG (each node's representative is itself)
2. For each $i \in [1, m]$, process equality $s_i = t_i$ as described

Full Algorithm for Deciding Satisfiability

Algorithm to decide satisfiability of $T_=$ formula

$$F: \; s_1 = t_1 \wedge \ldots s_m = t_m \wedge s_{m+1} \neq t_{m+1} \wedge \ldots s_n \neq t_n$$

1. Compute subterms and construct initial DAG (each node's representative is itself)
2. For each $i \in [1, m]$, process equality $s_i = t_i$ as described
3. For each $i \in [m+1, n]$, check if $Rep(s_i) = Rep(t_i)$

Full Algorithm for Deciding Satisfiability

Algorithm to decide satisfiability of $T_=$ formula

$$F: \; s_1 = t_1 \wedge \ldots s_m = t_m \wedge s_{m+1} \neq t_{m+1} \wedge \ldots s_n \neq t_n$$

1. Compute subterms and construct initial DAG (each node's representative is itself)
2. For each $i \in [1, m]$, process equality $s_i = t_i$ as described
3. For each $i \in [m+1, n]$, check if $Rep(s_i) = Rep(t_i)$
4. If there exists some $i \in [m+1, n]$ for which $Rep(s_i) = Rep(t_i)$, return UNSAT

Full Algorithm for Deciding Satisfiability

Algorithm to decide satisfiability of $T_=$ formula

$$F: \; s_1 = t_1 \wedge \ldots s_m = t_m \wedge s_{m+1} \neq t_{m+1} \wedge \ldots s_n \neq t_n$$

1. Compute subterms and construct initial DAG (each node's representative is itself)

2. For each $i \in [1, m]$, process equality $s_i = t_i$ as described

3. For each $i \in [m+1, n]$, check if $Rep(s_i) = Rep(t_i)$

4. If there exists some $i \in [m+1, n]$ for which $Rep(s_i) = Rep(t_i)$, return UNSAT

5. If for all i, $Rep(s_i) \neq Rep(t_i)$, return SAT

Example

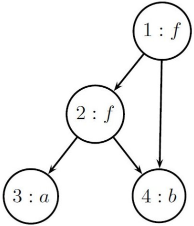
- Consider formula $F : f(a, b) = a \land f(f(a, b), b) \neq a$

Example

- Consider formula $F : f(a, b) = a \wedge f(f(a, b), b) \neq a$
- Subterms: $a, b, f(a, b), f(f(a, b), b)$

Example

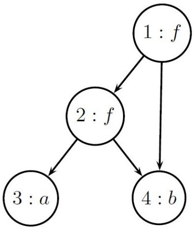
- Consider formula $F : f(a, b) = a \wedge f(f(a, b), b) \neq a$
- Subterms: $a, b, f(a, b), f(f(a, b), b)$



- Construct initial DAG

Example

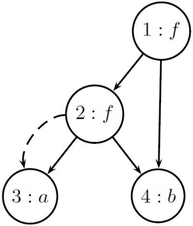
- Consider formula $F : f(a,b) = a \wedge f(f(a,b),b) \neq a$
- Subterms: $a, b, f(a,b), f(f(a,b),b)$



- Construct initial DAG
- Process equality $f(a,b) = a$

Example

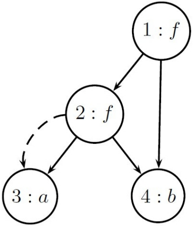
- Consider formula $F : f(a, b) = a \wedge f(f(a, b), b) \neq a$
- Subterms: $a, b, f(a, b), f(f(a, b), b)$



- Construct initial DAG
- Process equality $f(a, b) = a$

Example

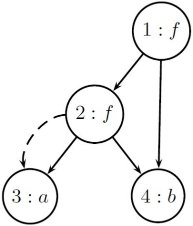
- Consider formula $F : f(a, b) = a \wedge f(f(a, b), b) \neq a$
- Subterms: $a, b, f(a, b), f(f(a, b), b)$



- Construct initial DAG
- Process equality $f(a, b) = a$
- Are parents $f(a, b)$ and $f(f(a, b), b)$ congruent?

Example

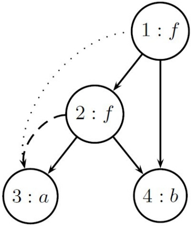
- Consider formula $F : f(a, b) = a \wedge f(f(a, b), b) \neq a$
- Subterms: $a, b, f(a, b), f(f(a, b), b)$



- Construct initial DAG
- Process equality $f(a, b) = a$
- Are parents $f(a, b)$ and $f(f(a, b), b)$ congruent?
- Yes, so process equality $f(a, b) = f(f(a, b), b)$

Example

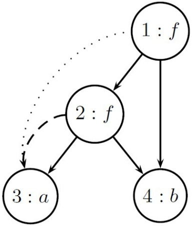
- Consider formula $F : f(a, b) = a \wedge f(f(a, b), b) \neq a$
- Subterms: $a, b, f(a, b), f(f(a, b), b)$



- Construct initial DAG
- Process equality $f(a, b) = a$
- Are parents $f(a, b)$ and $f(f(a, b), b)$ congruent?
- Yes, so process equality $f(a, b) = f(f(a, b), b)$

Example

- Consider formula $F : f(a, b) = a \wedge f(f(a, b), b) \neq a$
- Subterms: $a, b, f(a, b), f(f(a, b), b)$



- Construct initial DAG
- Process equality $f(a, b) = a$
- Are parents $f(a, b)$ and $f(f(a, b), b)$ congruent?
- Yes, so process equality $f(a, b) = f(f(a, b), b)$

- Formula unsatisfiable because $f(f(a, b), b)$ and a have same representative!

Example II

- Consider formula: $F : f^3(a) = a \wedge f^5(a) = a \wedge f(a) \neq a$

Example II

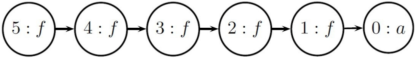
- Consider formula: $F : f^3(a) = a \land f^5(a) = a \land f(a) \neq a$

- Initial DAG:

$$5:f \rightarrow 4:f \rightarrow 3:f \rightarrow 2:f \rightarrow 1:f \rightarrow 0:a$$

Example II

- Consider formula: $F : f^3(a) = a \land f^5(a) = a \land f(a) \neq a$
- Initial DAG:

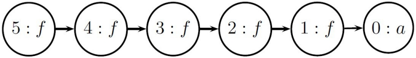


- Process equality $f^3(a) = a$:

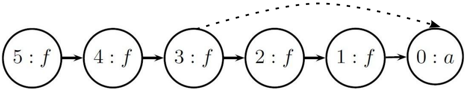
Example II

- Consider formula: $F : f^3(a) = a \wedge f^5(a) = a \wedge f(a) \neq a$

- Initial DAG:

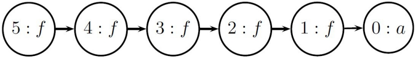


- Process equality $f^3(a) = a$:

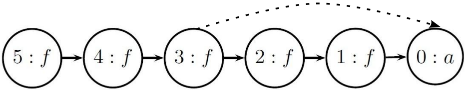


Example II

- Consider formula: $F : f^3(a) = a \wedge f^5(a) = a \wedge f(a) \neq a$
- Initial DAG:



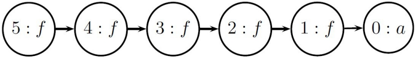
- Process equality $f^3(a) = a$:



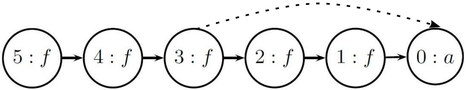
- Are parents congruent?

Example II

- Consider formula: $F : f^3(a) = a \wedge f^5(a) = a \wedge f(a) \neq a$
- Initial DAG:



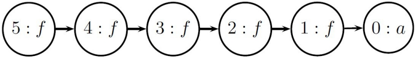
- Process equality $f^3(a) = a$:



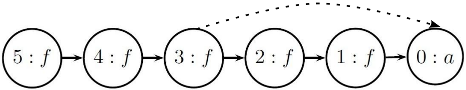
- Are parents congruent? Yes

Example II

- Consider formula: $F : f^3(a) = a \wedge f^5(a) = a \wedge f(a) \neq a$
- Initial DAG:



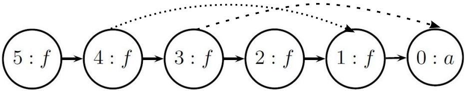
- Process equality $f^3(a) = a$:



- Are parents congruent? Yes
- Process equality $f^4(a) = f(a)$

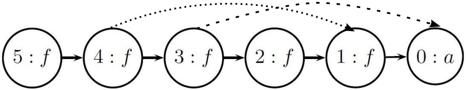
Example II, cont

- After merging classes:



Example II, cont

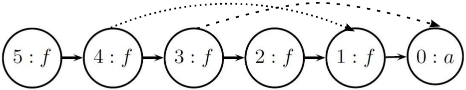
- After merging classes:



- Are $f^4(a)$'s and $f(a)$'s parents congruent?

Example II, cont

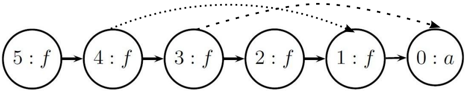
- After merging classes:



- Are $f^4(a)$'s and $f(a)$'s parents congruent? Yes

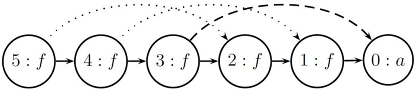
Example II, cont

- After merging classes:



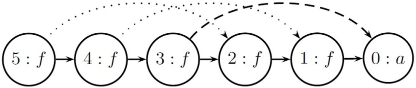
- Are $f^4(a)$'s and $f(a)$'s parents congruent? Yes

- Process equality $f^5(a) = f^2(a)$



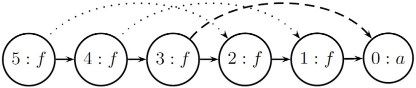
Example II, cont

- Formula: $F : f^3(a) = a \wedge f^5(a) = a \wedge f(a) \neq a$



Example II, cont

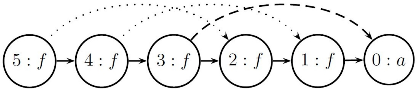
- Formula: $F : f^3(a) = a \wedge f^5(a) = a \wedge f(a) \neq a$



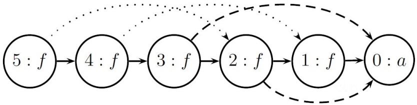
- Process equality $f^5(a) = a$:

Example II, cont

- Formula: $F : f^3(a) = a \wedge f^5(a) = a \wedge f(a) \neq a$

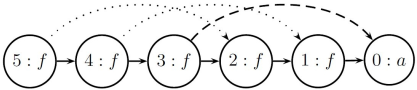


- Process equality $f^5(a) = a$:

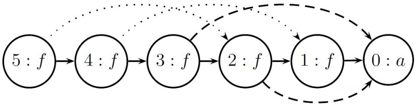


Example II, cont

- Formula: $F : f^3(a) = a \wedge f^5(a) = a \wedge f(a) \neq a$



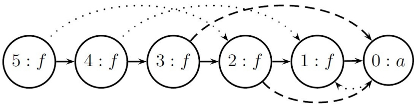
- Process equality $f^5(a) = a$:



- Now, parents $f^3(a)$ and $f(a)$ congruent; process equality $f^3(a) = f(a)$

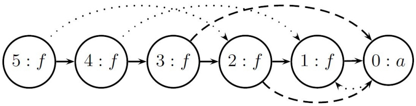
Example II, cont

- Formula: $F : f^3(a) = a \wedge f^5(a) = a \wedge f(a) \neq a$



Example II, cont

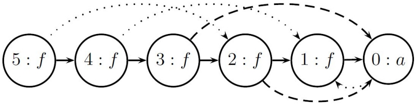
- Formula: $F : f^3(a) = a \wedge f^5(a) = a \wedge f(a) \neq a$



- Now, everything in same congruence class; so we are done.

Example II, cont

- Formula: $F : f^3(a) = a \wedge f^5(a) = a \wedge f(a) \neq a$



- Now, everything in same congruence class; so we are done.
- Formula UNSAT because a and $f(a)$ have same representative

Summary

- Congruence closure algorithm is used for determining satisfiability of $T_=$ formulas (without disjunction)

Summary

- Congruence closure algorithm is used for determining satisfiability of $T_=$ formulas (without disjunction)
- Our algorithm for computing congruence closures is called Union-Find, also used in other applications

Summary

- Congruence closure algorithm is used for determining satisfiability of $T_=$ formulas (without disjunction)
- Our algorithm for computing congruence closures is called Union-Find, also used in other applications
- Deciding conjuctive $T_=$ formulas is inexpensive: our algorithm is $O(e^2)$, but can be solved in $O(e\ log(e))$

Summary

- Congruence closure algorithm is used for determining satisfiability of $T_=$ formulas (without disjunction)
- Our algorithm for computing congruence closures is called Union-Find, also used in other applications
- Deciding conjuctive $T_=$ formulas is inexpensive: our algorithm is $O(e^2)$, but can be solved in $O(e\ log(e))$
- To decide satisfiability of formulas containing disjunctions, can either convert to DNF or use $\mathrm{DPLL}(\mathcal{T})$ (more on this later)

Summary

- Congruence closure algorithm is used for determining satisfiability of $T_=$ formulas (without disjunction)
- Our algorithm for computing congruence closures is called Union-Find, also used in other applications
- Deciding conjuctive $T_=$ formulas is inexpensive: our algorithm is $O(e^2)$, but can be solved in $O(e\ log(e))$
- To decide satisfiability of formulas containing disjunctions, can either convert to DNF or use $\mathrm{DPLL}(\mathcal{T})$ (more on this later)
- Next lecture: Decision procedure for qff theory of rationals (Simplex algorithm)

Summary

- Congruence closure algorithm is used for determining satisfiability of $T_=$ formulas (without disjunction)
- Our algorithm for computing congruence closures is called Union-Find, also used in other applications
- Deciding conjuctive $T_=$ formulas is inexpensive: our algorithm is $O(e^2)$, but can be solved in $O(e\ log(e))$
- To decide satisfiability of formulas containing disjunctions, can either convert to DNF or use $\mathrm{DPLL}(\mathcal{T})$ (more on this later)
- Next lecture: Decision procedure for qff theory of rationals (Simplex algorithm)
- Reminder: Homework due next lecture!!