SMT: Equality Logic With Uninterpreted Functions

Lukas Koller

June 20, 2020

Program Verification: OpenJML Example

- `https://www.rise4fun.com/OpenJMLESC/BinarySearch`

Program Verification: OpenJML Example

- How does OpenJML work?

Code + Pre- & Post- conditions $\xrightarrow{\text{OpenJML}}$ Formula (SMT-LIB) $\xrightarrow{\text{SMT solver (Z3, CVC4, \ldots)}}$ SATISFIABLE/ UNSATISFIABLE

Program Verification: OpenJML Example

- How does OpenJML work?

Code + Pre- & Post-conditions $\xrightarrow{\text{OpenJML}}$ Formula (SMT-LIB) $\xrightarrow{\text{SMT solver (Z3, CVC4, \ldots)}}$ SATISFIABLE/ UNSATISFIABLE

How do SMT solvers reason about equality?

What is SMT?

What is SMT?

- Satisfiability Modulo Theories (SMT) is a generalization of the Boolean satisfiability problem (SAT) for first-order logic

First-Order Theories

First-Order Theories

- A first-order theory is an extension of Boolean logic with specific constants, predicates, functions, and quantifiers

First-Order Theories

- A first-order theory is an extension of Boolean logic with specific constants, predicates, functions, and quantifiers

Example (First-Order Theories)

- Theory of Equality Logic: $x_1 \neq x_2 \wedge x_1 = 4$
- Theory of Linear Arithmetic: $(3x^2 + 2x - 1 = 0) \wedge (0 < x)$
- Theory of Bit Vectors: $(a \gg 2) = c \wedge c \oplus d$

How do SMT solvers reason about equality?

How do SMT solvers reason about equality?

- Solver for Theory of Equality Logic With Uninterpreted Functions (EUF)

Theory of Equality Logic With Uninterpreted Functions (EUF)

Theory of Equality Logic With Uninterpreted Functions (EUF)

- Introduces the equality predicate $(=)$

$$
\begin{aligned}
&\forall x.\ x = x && \text{(Reflexivity)}\\
&\forall x.\ \forall y.\ x = y \implies y = x && \text{(Symmetry)}\\
&\forall x.\ \forall y.\ \forall z.\ x = y \land y = z \implies x = z && \text{(Transitivity)}
\end{aligned}
$$

Theory of Equality Logic With Uninterpreted Functions (EUF)

- Introduces the equality predicate $(=)$

$$\forall x.\ x = x \qquad (\textsc{Reflexivity})$$

$$\forall x.\ \forall y.\ x = y \implies y = x \qquad (\textsc{Symmetry})$$

$$\forall x.\ \forall y.\ \forall z.\ x = y \land y = z \implies x = z \qquad (\textsc{Transitivity})$$

- Variables are defined over an infinite domain, such as $\mathbb{N}$ or $\mathbb{R}$

Theory of Equality Logic With Uninterpreted Functions (EUF)

- Introduces the equality predicate $(=)$

$$\forall x.\ x = x \qquad (\textsc{Reflexivity})$$
$$\forall x.\ \forall y.\ x = y \implies y = x \qquad (\textsc{Symmetry})$$
$$\forall x.\ \forall y.\ \forall z.\ x = y \land y = z \implies x = z \qquad (\textsc{Transitivity})$$

- Variables are defined over an infinite domain, such as $\mathbb{N}$ or $\mathbb{R}$
- Functions are uninterpreted $\implies$ only functional congruence

Theory of Equality Logic With Uninterpreted Functions (EUF)

- Introduces the equality predicate $(=)$

$$\forall x.\ x = x \qquad \text{(Reflexivity)}$$
$$\forall x.\ \forall y.\ x = y \implies y = x \qquad \text{(Symmetry)}$$
$$\forall x.\ \forall y.\ \forall z.\ x = y \land y = z \implies x = z \qquad \text{(Transitivity)}$$

- Variables are defined over an infinite domain, such as $\mathbb{N}$ or $\mathbb{R}$
- Functions are uninterpreted $\implies$ only functional congruence

Example (Formula in EUF)

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

Functional Congruence

Definition (Functional Congruence)

For each $n > 0$ and n-ary function f

$$\forall \bar{x}, \bar{y}.\ \bigwedge_{i=1}^{n} x_i = y_i \implies f(\bar{x}) = f(\bar{y})$$

- Ignore details and characteristics of a function

Functional Congruence

Example (Uninterpreted Functions & Commutativity)

The following formula is valid as $+$ is commutative.

$$x_1 = y_1 \land x_2 = y_2 \implies x_1 + x_2 = y_2 + y_1$$

Functional Congruence

Example (Uninterpreted Functions & Commutativity)

The following formula is valid as $+$ is commutative.

$$x_1 = y_1 \wedge x_2 = y_2 \implies x_1 + x_2 = y_2 + y_1$$

Abstracting $+$ with an uninterpreted function symbol f:

$$x_1 = y_1 \wedge x_2 = y_2 \implies f(x_1, x_2) = f(y_2, y_1)$$

Functional Congruence

Example (Uninterpreted Functions & Commutativity)

The following formula is valid as $+$ is commutative.

$$x_1 = y_1 \land x_2 = y_2 \implies x_1 + x_2 = y_2 + y_1$$

Abstracting $+$ with an uninterpreted function symbol f:

$$x_1 = y_1 \land x_2 = y_2 \implies f(x_1, x_2) = f(y_2, y_1)$$

$\rightarrow$ Commutativity is lost

Functional Congruence

Example (Uninterpreted Functions & Commutativity)

The following formula is valid as $+$ is commutative.

$$x_1 = y_1 \wedge x_2 = y_2 \implies x_1 + x_2 = y_2 + y_1$$

Abstracting $+$ with an uninterpreted function symbol f:

$$x_1 = y_1 \wedge x_2 = y_2 \implies f(x_1, x_2) = f(y_2, y_1)$$

$\rightarrow$ Commutativity is lost

Additional constraint to keep the commutativity:

$$x_1 = y_1 \wedge x_2 = y_2 \implies f(x_1, x_2) = f(y_2, y_1) \vee f(x_1, x_2) = f(y_1, y_2)$$

Congruence Closure Algorithm

Congruence Closure Algorithm

- Satisfiability of a conjunction of equalities and inequalities with uninterpreted functions

Congruence Closure Algorithm

- Satisfiability of a conjunction of equalities and inequalities with uninterpreted functions

Algorithm (Congruence Closure Algorithm)

$$F : (\bigwedge_{i=1}^{m} s_i = t_i) \wedge (\bigwedge_{j=m+1}^{n} s_j \neq t_j)$$

S: the set of all equalities and inequalities in F
T: set of all terms and subterms in F

A partition of T is constructed as follows:

(1) initial partition $\{\{t\} \mid t \in T\}$

(2) for all $1 \leq i \leq m$

  a. with $s_i = t_i$ merge the congruence classes of s_i and t_i
  b. propagate the new congruence with symmetry, transitivity, and functional congruence

Congruence Closure Algorithm

Constructed partition induces a *congruence relation* $\sim$ on T

- $\sim$ is reflexive, symmetric, & transitive (equivalence relation)
- $\sim$ respects functional congruence

Congruence Closure Algorithm

Constructed partition induces a *congruence relation* $\sim$ on T

- $\sim$ is reflexive, symmetric, & transitive (equivalence relation)
- $\sim$ respects functional congruence

Theorem

F is satisfiable $\iff \nexists s_i, t_i \in T$ such that $s_i \sim t_i$ and $(s_i \neq t_i) \in S$

Congruence Closure Algorithm: Example

Example (Congruence Closure Algorithm)

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

- initial partition:

$\{a\}$ $\{b\}$

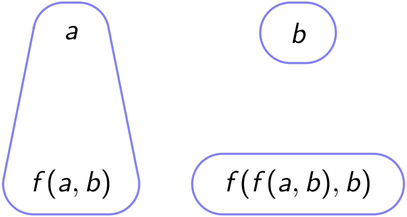
$\{f(a, b)\}$ $\{f(f(a, b), b)\}$

Congruence Closure Algorithm: Example

Example (Congruence Closure Algorithm)

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

- with $f(a, b) = a$ merge the congruence classes of $f(a, b)$ and a:

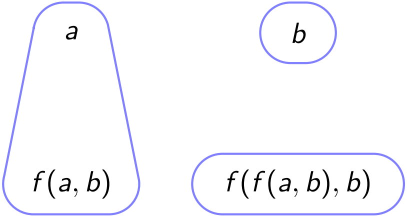


Congruence Closure Algorithm: Example

Example (Congruence Closure Algorithm)

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

- $a \sim f(a,b)$, with functional congruence $f(a,b) \sim f(f(a,b),b)$:

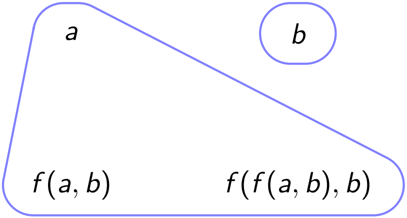


Congruence Closure Algorithm: Example

Example (Congruence Closure Algorithm)

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

- $a \sim f(a, b)$, with functional congruence $f(a, b) \sim f(f(a, b), b)$:

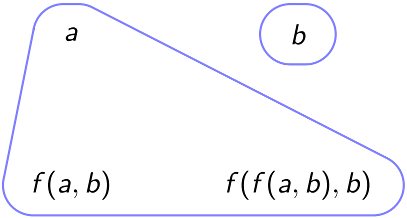


Congruence Closure Algorithm: Example

Example (Congruence Closure Algorithm)

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

- $a \sim f(a, b)$, with functional congruence $f(a, b) \sim f(f(a, b), b)$:



$\implies f(f(a, b), b) \sim a$, but $f(f(a, b), b) \neq a \implies F$ is UNSATISFIABLE

Satisfiability of Arbitrary EUF-Formulas

- Congruence Closure algorithm only for conjunctions of equalities and inequalities

Satisfiability of Arbitrary EUF-Formulas

- Congruence Closure algorithm only for conjunctions of equalities and inequalities

Algorithm (Satisfiability of Arbitrary EUF-Formulas)

1. Negate F & convert to DNF, yields F'
2. check unsatisfiability for each disjunct of F' with Congruence Closure algorithm
3. if all disjuncts of F' are unsatisfiable
   then return SATISFIABLE
   else return UNSATISFIABLE

Satisfiability of Arbitrary EUF-Formulas

- Congruence Closure algorithm only for conjunctions of equalities and inequalities

Algorithm (Satisfiability of Arbitrary EUF-Formulas)

1. Negate F & convert to DNF, yields F'
2. check unsatisfiability for each disjunct of F' with Congruence Closure algorithm
3. if all disjuncts of F' are unsatisfiable
   then return SATISFIABLE
   else return UNSATISFIABLE

possible exponential blowup of DNF

Efficient Implementation of the Congruence Closure Algorithm

Constructed congruence relation $\sim$ on T

- $\sim$ is reflexive, symmetric, & transitive (equivalence relation)
- $\sim$ respects functional congruence

$\Longrightarrow$ Idea: Union-Find algorithm + efficient propagation of functional congruence

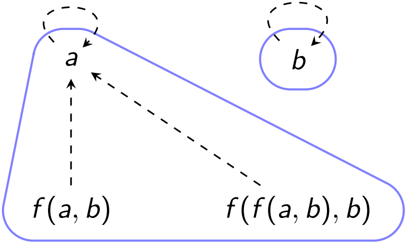
Efficient Implementation of the Congruence Closure Algorithm: Union-Find

General idea

- Membership to an equivalence class represented by reference to representative element

Example (Union-Find)

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



How to efficiently propagate functional congruence?

How to efficiently propagate functional congruence?

$$\forall \bar{x}, \bar{y}. \bigwedge_{i=1}^{n} x_i = y_i \implies f(\bar{x}) = f(\bar{y})$$

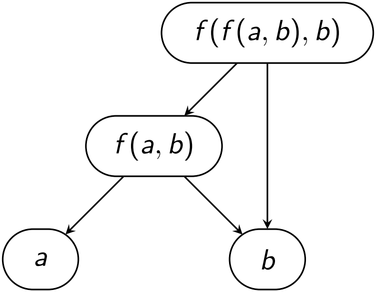
$\implies$ functional congruence is only propagated from function arguments to function applications

$\implies$ *Directed-Acyclic-Graph* (DAG)

Efficient Implementation of the Congruence Closure Algorithm: DAG

Example (DAG)

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



$\implies$ efficient propagation of functional congruence to predecessors

Efficient Implementation of the Congruence Closure Algorithm: Propagation of Functional Congruence

- P_s: all predecessors the congruence class that contains s
- P_t: all predecessors the congruence class that contains t
- new congruence $s \sim t$
- for any $(s', t') \in P_s \times P_t$ check if $s' \overset{?}{\sim} t'$

Efficient Implementation of the Congruence Closure Algorithm: Union-Find + DAG

Example (Union-Find + DAG)

$$G : f^5(a) = a \wedge f^3(a) = a \wedge f(a) \neq a$$

- DAG for G:

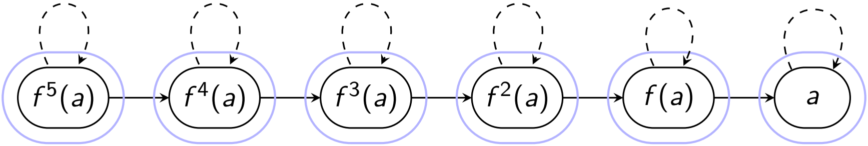
$$f^5(a) \rightarrow f^4(a) \rightarrow f^3(a) \rightarrow f^2(a) \rightarrow f(a) \rightarrow a$$

Efficient Implementation of the Congruence Closure Algorithm: Union-Find + DAG

Example (Union-Find + DAG)

$$G : f^5(a) = a \wedge f^3(a) = a \wedge f(a) \neq a$$

- Initial partition:

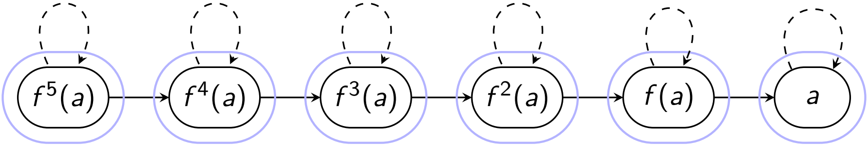


Efficient Implementation of the Congruence Closure Algorithm: Union-Find + DAG

Example (Union-Find + DAG)

$$G : f^5(a) = a \land f^3(a) = a \land f(a) \neq a$$

- $f^5(a) = a \implies f^5(a) \sim a$



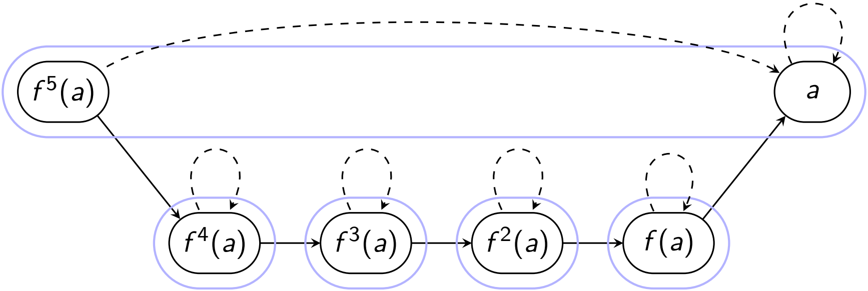
$P_{f^5(a)} \times P_a = \{\}$

Efficient Implementation of the Congruence Closure Algorithm: Union-Find + DAG

Example (Union-Find + DAG)

$$G : f^5(a) = a \land f^3(a) = a \land f(a) \neq a$$

- $f^5(a) = a \implies f^5(a) \sim a$



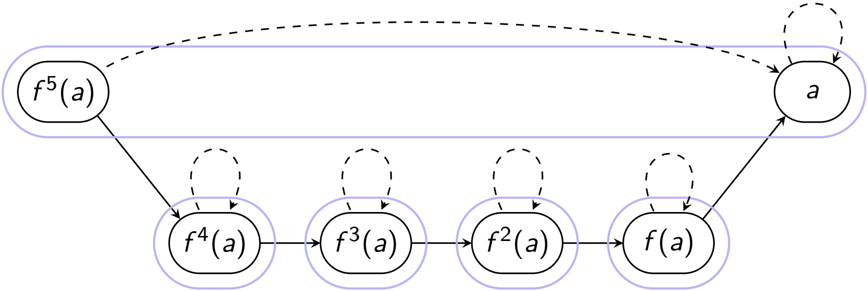
$P_{f^5(a)} \times P_a = \{\}$

Efficient Implementation of the Congruence Closure Algorithm: Union-Find + DAG

Example (Union-Find + DAG)

$$G : f^5(a) = a \wedge f^3(a) = a \wedge f(a) \neq a$$

- $f^3(a) = a \implies f^3(a) \sim a$



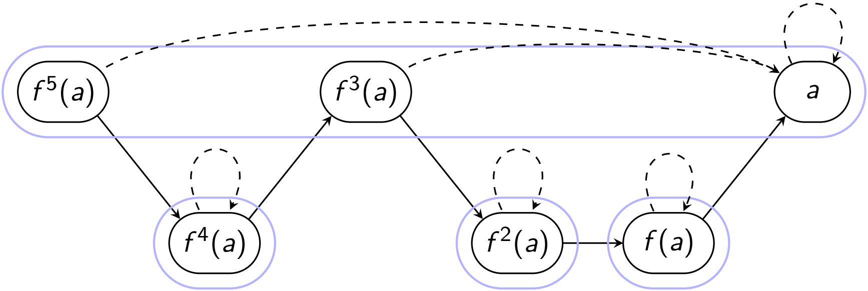
$P_{f^3(a)} \times P_a = \{(f^4(a), f(a))\}$

Efficient Implementation of the Congruence Closure Algorithm: Union-Find + DAG

Example (Union-Find + DAG)

$$G : f^5(a) = a \wedge f^3(a) = a \wedge f(a) \neq a$$

- $f^3(a) = a \implies f^3(a) \sim a$



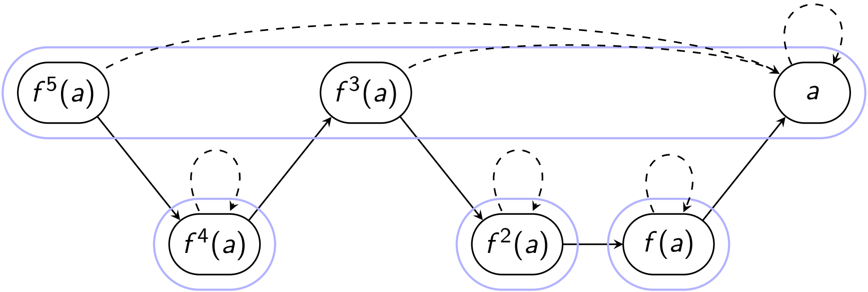
$$P_{f^3(a)} \times P_a = \{(f^4(a), f(a))\} \implies f^4(a) \overset{?}{\sim} f(a)$$

Efficient Implementation of the Congruence Closure Algorithm: Union-Find + DAG

Example (Union-Find + DAG)

$$G : f^5(a) = a \wedge f^3(a) = a \wedge f(a) \neq a$$

- $f^4(a) \overset{?}{\sim} f(a)$

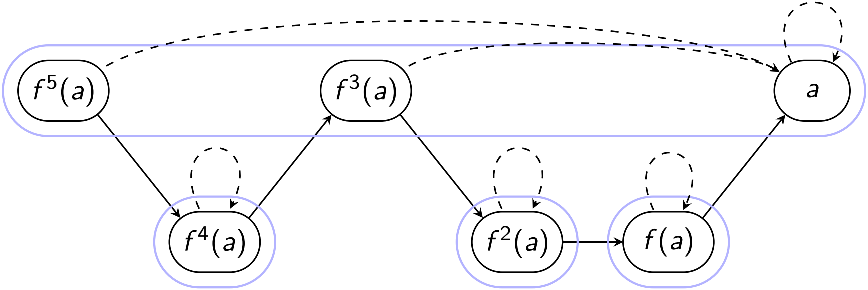


Efficient Implementation of the Congruence Closure Algorithm: Union-Find + DAG

Example (Union-Find + DAG)

$$G : f^5(a) = a \land f^3(a) = a \land f(a) \neq a$$

- with functional congruence $f^4(a) \sim f(a)$



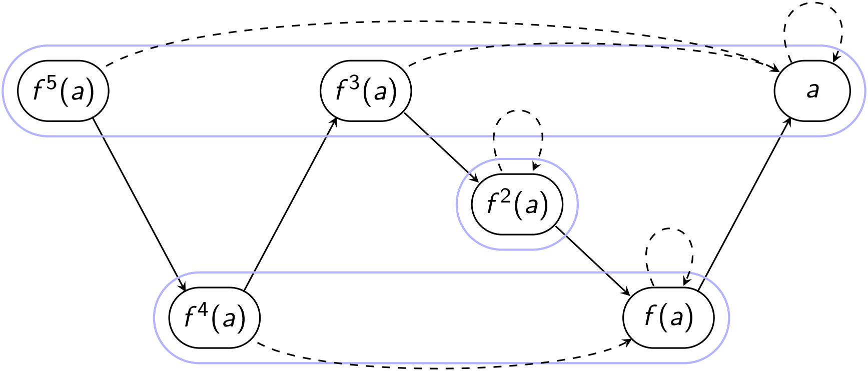
$P_{f^4(a)} \times P_{f(a)} = \{(f^5(a), f^2(a))\}$

Efficient Implementation of the Congruence Closure Algorithm: Union-Find + DAG

Example (Union-Find + DAG)

$$G : f^5(a) = a \land f^3(a) = a \land f(a) \neq a$$

- with functional congruence $f^4(a) \sim f(a)$



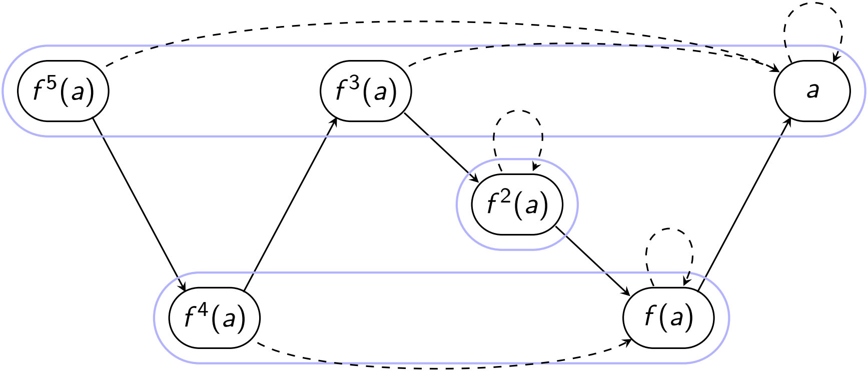
$P_{f^4(a)} \times P_{f(a)} = \{(f^5(a), f^2(a))\} \implies f^5(a) \overset{?}{\sim} f^2(a)$

Efficient Implementation of the Congruence Closure Algorithm: Union-Find + DAG

Example (Union-Find + DAG)

$$G : f^5(a) = a \land f^3(a) = a \land f(a) \neq a$$

- $f^5(a) \overset{?}{\sim} f^2(a)$

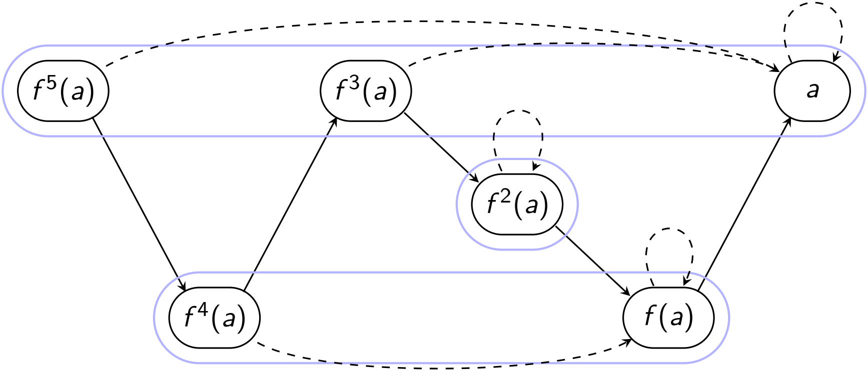


Efficient Implementation of the Congruence Closure Algorithm: Union-Find + DAG

Example (Union-Find + DAG)

$$G : f^5(a) = a \wedge f^3(a) = a \wedge f(a) \neq a$$

- with functional congruence $f^5(a) \sim f^2(a)$



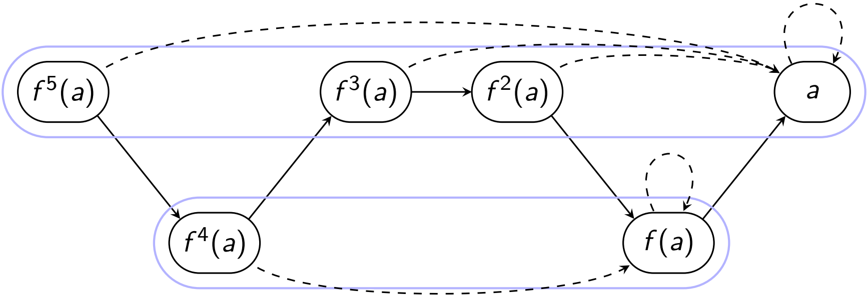
$$P_{f^5(a)} \times P_{f^2(a)} = \{(f^4(a), f^3(a)), (f(a), f^3(a))\}$$

Efficient Implementation of the Congruence Closure Algorithm: Union-Find + DAG

Example (Union-Find + DAG)

$$G : f^5(a) = a \land f^3(a) = a \land f(a) \neq a$$

- with functional congruence $f^5(a) \sim f^2(a)$



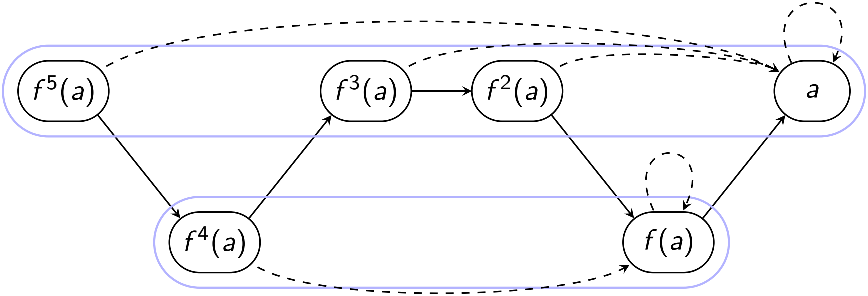
$P_{f^5(a)} \times P_{f^2(a)} = \{(f^4(a), f^3(a)), (f(a), f^3(a))\}$
$\implies f^4(a) \overset{?}{\sim} f^3(a)$ and $f(a) \overset{?}{\sim} f^3(a)$

Efficient Implementation of the Congruence Closure Algorithm: Union-Find + DAG

Example (Union-Find + DAG)

$$G : f^5(a) = a \land f^3(a) = a \land f(a) \neq a$$

- $f^4(a) \overset{?}{\sim} f^3(a)$

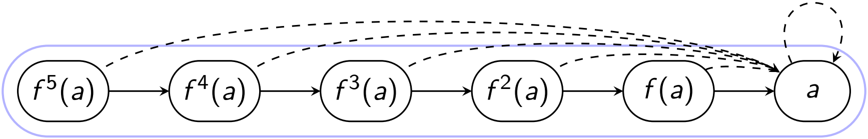


Efficient Implementation of the Congruence Closure Algorithm: Union-Find + DAG

Example (Union-Find + DAG)

$$G : f^5(a) = a \land f^3(a) = a \land f(a) \neq a$$

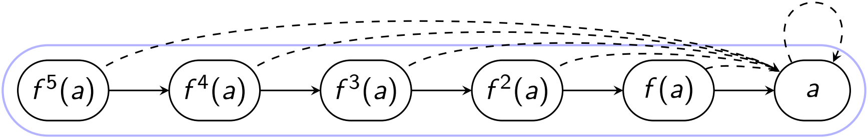
- $f^4(a) \sim f^3(a)$



Efficient Implementation of the Congruence Closure Algorithm: Union-Find + DAG

Example (Union-Find + DAG)

$$G : f^5(a) = a \wedge f^3(a) = a \wedge f(a) \neq a$$



$\implies f(a) \sim a$, but $f(a) \neq a \implies G$ is UNSATISFIABLE

Conclusion

- Fast implementations of the Congruence Closure algorithm take $\mathcal{O}(n \log n)$ time, for formulas of size n

Conclusion

- Fast implementations of the Congruence Closure algorithm take $\mathcal{O}(n \log n)$ time, for formulas of size n
- Any modern SMT Solver contains an efficient solver for EUF

Conclusion

- Fast implementations of the Congruence Closure algorithm take $\mathcal{O}(n \log n)$ time, for formulas of size n
- Any modern SMT Solver contains an efficient solver for EUF
- SMT solvers have numerous applications
  - Program & hardware verification $\implies$ OpenJML
  - Testcase-generation
  - Static analysis
  - ...

$\implies$ in almost all applications a SMT solver must be able to efficiently reason about equality