Automated Reasoning and Program Verification

Laura Kovács
TU Vienna
Outline

Theory of Equality

Theory of Equality: Congruence Closure
Congruence Closure and DAGs
The **theory of equality** $\mathcal{T}_E$ is defined by

- a signature $\Sigma_E = \{a, b, \ldots, f, g, \ldots, =, p, \ldots\}$
- the previously given five axioms, that is:

  \[
  \begin{align*}
  x &= x & \text{(reflexivity)} \\
  x = y & \rightarrow y = x & \text{(symmetry)} \\
  x = y \land y = z & \rightarrow x = z & \text{(transitivity)} \\
  x_1 = y_1 \land \ldots \land x_n = y_n & \rightarrow f(x_1, \ldots, x_n) = f(y_1, \ldots, y_n) & \text{(function congruence)} \\
  x_1 = y_1 \land \ldots \land x_n = y_n \land p(x_1, \ldots, x_n) & \rightarrow p(y_1, \ldots, y_n) & \text{(predicate congruence)}
  \end{align*}
  \]
A Satisfiability Question in the Theory of Equality

- Deciding a conjunction of $T_E$-literals: How can we check whether a set of $T_E$-litersals is satisfiable?
Outline

Theory of Equality

Theory of Equality: Congruence Closure
Congruence Closure and DAGs
Deciding $\mathcal{T}_E$: An Example

Question: Is $a = b \wedge b = c \wedge f(a) \neq f(c)$ satisfiable in $\mathcal{T}_E$?
Deciding $\mathcal{T}_E$: An Example

**Question:** Is $a = b \land b = c \land f(a) \neq f(c)$ satisfiable in $\mathcal{T}_E$?

- From $a = b \land b = c$ and (transitivity), conclude $a = c$.
- From $a = c$ and (congruence), conclude $f(a) = f(c)$.
- $f(a) = f(c)$ contradicts $f(a) \neq f(c)$.
Deciding $\mathcal{T}_E$: An Example

**Question**: Is $a = b \land b = c \land f(a) \neq f(c)$ satisfiable in $\mathcal{T}_E$?

- From $a = b \land b = c$ and (transitivity), conclude $a = c$.
- From $a = c$ and (congruence), conclude $f(a) = f(c)$.
- $f(a) = f(c)$ contradicts $f(a) \neq f(c)$.

**Question**: Is $x = y \land f(f(x)) \neq f(f(y))$ satisfiable in $\mathcal{T}_E$?
Deciding $\mathcal{T}_E$: An Example

Question: Is $a = b \land b = c \land f(a) \neq f(c)$ satisfiable in $\mathcal{T}_E$?

- From $a = b \land b = c$ and (transitivity), conclude $a = c$.
- From $a = c$ and (congruence), conclude $f(a) = f(c)$.
- $f(a) = f(c)$ contradicts $f(a) \neq f(c)$.

Question: Is $x = y \land f(f(x)) \neq f(f(y))$ satisfiable in $\mathcal{T}_E$?

The reasoning made above is very different from splitting or DPLL. It uses theory axioms.
Deciding $\mathcal{T}_E$: An Example

Question: Is $a = b \land b = c \land f(a) \neq f(c)$ satisfiable in $\mathcal{T}_E$?

- From $a = b \land b = c$ and (transitivity), conclude $a = c$.
- From $a = c$ and (congruence), conclude $f(a) = f(c)$.
- $f(a) = f(c)$ contradicts $f(a) \neq f(c)$.

Question: Is $x = y \land f(f(x)) \neq f(f(y))$ satisfiable in $\mathcal{T}_E$?

The reasoning made above is very different from splitting or DPLL. It uses theory axioms.

We will now discuss specialised decision procedures for theories.
Deciding $\mathcal{T}_E$: Congruence Closure

Congruence closure

- is a method to decide satisfiability of formulas in $\mathcal{T}_E$;

What about formulas with predicates other than equality?

Formulas with uninterpreted predicates can be easily transformed to formulas without predicates other than $\mathcal{E}$.

Example:

Instead of $p(x) \land q(y, z) \land a = b \rightarrow \neg q(x, z)$

we use $f_p(x) = t \land f_q(y, z) = t \land a = b \rightarrow f_q(x, z) \neq t$,

where $f_p$, $f_q$ are fresh functions and $t$ is a fresh constant.
Deciding $\mathcal{T}_E$: Congruence Closure

Congruence closure

- is a method to decide satisfiability of formulas in $\mathcal{T}_E$;
- a decision procedure for $\mathcal{T}_E$;
Deciding $\mathcal{T}_E$: Congruence Closure

Congruence closure
- is a method to decide satisfiability of formulas in $\mathcal{T}_E$;
- a decision procedure for $\mathcal{T}_E$;
- can be extended to a decision procedure for $\mathcal{T}_A$;
- is the basis for combining theories;

What about formulas with predicates other than equality? Formulas with uninterpreted predicates can be easily transformed to formulas without predicates other than $\mathcal{E}$. Example:

Instead of $p(x) \land q(y, z) \land a = b \rightarrow \neg q(x, z)$
we use $f(p(x)) = t \land f(q(y, z)) = t \land a = b \rightarrow f(q(x, z)) \neq t$,
where $f(p), f(q)$ are fresh functions and $t$ is a fresh constant.
Deciding $\mathcal{T}_E$: Congruence Closure

Congruence closure

- is a method to decide satisfiability of formulas in $\mathcal{T}_E$;
- a decision procedure for $\mathcal{T}_E$;

- decides formulas in the theory of equality and uninterpreted functions.

What about formulas with predicates other than equality?

Formulas with uninterpreted predicates can be easily transformed to formulas without predicates other than $=$.

Example: Instead of $p(x) \land q(y, z) \land a = b \rightarrow \neg q(x, z)$ we use $f(p)(x) = t \land f(q)(y, z) = t \land a = b \rightarrow f(q)(x, z) \neq t$, where $f_p, f_q$ are fresh functions and $t$ is a fresh constant.
Deciding $\mathcal{T}_E$: Congruence Closure

Congruence closure

- is a method to decide satisfiability of formulas in $\mathcal{T}_E$;
- a decision procedure for $\mathcal{T}_E$;
- decides formulas in the theory of equality and uninterpreted functions.

What about formulas with predicates other than equality?
Deciding $\mathcal{T}_E$: Congruence Closure

Congruence closure

- is a method to decide satisfiability of formulas in $\mathcal{T}_E$;
- a decision procedure for $\mathcal{T}_E$;

- decides formulas in the theory of equality and uninterpreted functions.

What about formulas with predicates other than equality?

Formulas with uninterpreted predicates can be easily transformed to formulas without predicates other than $=$.

Example: $p(x) \land q(y, z) \land a = b \rightarrow \neg q(x, z)$
Deciding $\mathcal{T}_E$: Congruence Closure

Congruence closure
- is a method to decide satisfiability of formulas in $\mathcal{T}_E$;
- a decision procedure for $\mathcal{T}_E$;
- decides formulas in the theory of equality and uninterpreted functions.

What about formulas with predicates other than equality?

Formulas with uninterpreted predicates can be easily transformed to formulas without predicates other than $=$.

Example: Instead of $p(x) \land q(y, z) \land a = b \rightarrow \neg q(x, z)$
we use

$$f_p(x) = t \land f_q(y, z) = t \land a = b \rightarrow f_q(x, z) \neq t,$$

where $f_p, f_q$ are fresh functions and $t$ is a fresh constant.
Deciding $\mathcal{T}_E$: Congruence Closure – abstract definitions

Consider a set $S$ and a binary relation $R$ over $S$. $R$ is a congruence relation if:

- $xRx$
- $xRy \rightarrow yRx$
- $xRy \land yRz \rightarrow xRz$
- $x_1Ry_1 \land \ldots \land x_nRy_n \rightarrow f(x_1, \ldots, x_n)Rf(y_1, \ldots, y_n)$ for all function symbols $f$.

Example: $=$ is a congruence relation.

The congruence class of $t \in S$ under the congruence relation $R$ is $[t]_R = \{ t' \in S | tRt' \}$.

The congruence relation $R$ defines a partition on $S$: $(\bigcup [t]_R) = S$ and $[t_1]_R \neq [t_2]_R \rightarrow [t_1]_R \cap [t_2]_R = \emptyset$. 
Deciding $T_E$: Congruence Closure – abstract definitions

Consider a set $S$ and a binary relation $R$ over $S$. $R$ is a congruence relation if:

- $xRx$
- $xRy \rightarrow yRx$
- $xRy \land yRz \rightarrow xRz$
- $x_1 R y_1 \land \ldots \land x_n R y_n \rightarrow f(x_1, \ldots, x_n) R f(y_1, \ldots, y_n)$ for all function symbols $f$.

Example: $=$ is a congruence relation.
Deciding $\mathcal{T}_E$: Congruence Closure – abstract definitions

Consider a set $S$ and a binary relation $R$ over $S$. $R$ is a congruence relation if:

- $xRx$
- $xRy \rightarrow yRx$
- $xRy \land yRz \rightarrow xRz$
- $x_1Ry_1 \land \ldots \land x_nRy_n \rightarrow f(x_1, \ldots, x_n)Rf(y_1, \ldots, y_n)$ for all function symbols $f$.

Example: $=$ is a congruence relation.

The congruence class of $t \in S$ under the congruence relation $R$ is

$$[t]_R = \{ t' \in S \mid tRt' \}$$
Deciding $\mathcal{T}_E$: Congruence Closure – abstract definitions

Consider a set $S$ and a binary relation $R$ over $S$. $R$ is a congruence relation if:

- $xRx$
- $xRy \rightarrow yRx$
- $xRy \land yRz \rightarrow xRz$
- $x_1Ry_1 \land \ldots \land x_nRy_n \rightarrow f(x_1, \ldots, x_n)Rf(y_1, \ldots, y_n)$ for all function symbols $f$.

Example: $=$ is a congruence relation.

The congruence class of $t \in S$ under the congruence relation $R$ is

$$[t]_R = \{ t' \in S \mid tRt' \}$$

The congruence relation $R$ defines a partition on $S$:

$$\bigcup_{[t]_R} [t]_R = S \quad \text{and} \quad [t_1]_R \neq [t_2]_R \rightarrow [t_1]_R \cap [t_2]_R = \emptyset$$
The congruence closure $R^c$ of $R$ is the congruence relation such that:

- $R \subseteq R^c$
- for all other congruence relations $R'$ with $R \subseteq R'$, either $R^c = R'$ or $R^c \subseteq R'$

$R^c$ is the smallest congruence relation that includes $R$. 

Example:

Let $S = \{a, b, c, d\}$ and $R = \{aRb, bRc, dRd\}$. Then $R^c = \{aRb, bRc, dRd, aRa, bRb, cRc, bRa, cRb, aRc, cRa\}$ since:
- $aRb, bRc, dRd \in R^c$ by $R \subseteq R^c$
- $aRa, bRb, cRc \in R^c$ by reflexivity
- $bRa, cRb \in R^c$ by symmetry
- $aRc \in R^c$ by transitivity
- $cRa \in R^c$ by symmetry
Deciding $\mathcal{T}_E$: Congruence Closure – abstract definitions

The congruence closure $R^c$ of $R$ is the congruence relation such that:

- $R \subseteq R^c$
- for all other congruence relations $R'$ with $R \subseteq R'$, either $R^c = R'$ or $R^c \subseteq R'$

$R^c$ is the smallest congruence relation that includes $R$.

**Example:** Let $S = \{a, b, c, d\}$ and $R = \{aRb, b Rc, dRd\}$. Then

$$R^c = \{aRb, b Rc, dRd, aRa, bRb, cRc, bRa, cRb, aRc, cRa\}$$
Deciding $\mathcal{T}_E$: Congruence Closure – abstract definitions

The congruence closure $R^c$ of $R$ is the congruence relation such that:

- $R \subseteq R^c$
- for all other congruence relations $R'$ with $R \subseteq R'$, either $R^c = R'$ or $R^c \subseteq R'$

$R^c$ is the smallest congruence relation that includes $R$.

**Example:** Let $S = \{a, b, c, d\}$ and $R = \{aRb, bRc, dRd\}$. Then

$$R^c = \{aRb, bRc, dRd, aRa, bRb, cRc, bRa, cRb, aRc, cRa\}$$ since

- $aRb, bRc, dRd \in R^c$ by $R \subseteq R^c$
Deciding $T_E$: Congruence Closure – abstract definitions

The congruence closure $R^c$ of $R$ is the congruence relation such that:

- $R \subseteq R^c$
- for all other congruence relations $R'$ with $R \subseteq R'$, either $R^c = R'$ or $R^c \subseteq R'$

$R^c$ is the smallest congruence relation that includes $R$.

Example: Let $S = \{a, b, c, d\}$ and $R = \{aRb, bRc, dRd\}$. Then

$$R^c = \{aRb, bRc, dRd, aRa, bRb, cRc, bRa, cRb, aRc, cRa\} \text{ since}$$

- $aRb, bRc, dRd \in R^c$ by $R \subseteq R^c$
- $aRa, bRb, cRc \in R^c$ by reflexivity
The congruence closure $R^c$ of $R$ is the congruence relation such that:

1. $R \subseteq R^c$
2. for all other congruence relations $R'$ with $R \subseteq R'$, either $R^c = R'$ or $R^c \subseteq R'$

$R^c$ is the smallest congruence relation that includes $R$.

**Example:** Let $S = \{a, b, c, d\}$ and $R = \{aRb, bRc, dRd\}$. Then

$$R^c = \{aRb, bRc, dRd, aRa, bRb, cRc, bRa, cRb, aRc, cRa\}$$ since

- $aRb, bRc, dRd \in R^c$ by $R \subseteq R^c$
- $aRa, bRb, cRc \in R^c$ by reflexivity
- $bRa, cRb \in R^c$ by symmetry
The congruence closure $R^c$ of $R$ is the congruence relation such that:

- $R \subseteq R^c$
- for all other congruence relations $R'$ with $R \subseteq R'$, either $R^c = R'$ or $R^c \subseteq R'$

$R^c$ is the smallest congruence relation that includes $R$.

**Example:** Let $S = \{a, b, c, d\}$ and $R = \{aRb, bRc, dRd\}$. Then

$$R^c = \{aRb, bRc, dRd, aRa, bRb, cRc, bRa, cRb, aRc, cRa\}$$ since

- $aRb, bRc, dRd \in R^c$ by $R \subseteq R^c$
- $aRa, bRb, cRc \in R^c$ by reflexivity
- $bRa, cRb \in R^c$ by symmetry
- $aRc \in R^c$ by transitivity
Deciding $\mathcal{T}_E$: Congruence Closure – abstract definitions

The congruence closure $R^c$ of $R$ is the congruence relation such that:

- $R \subseteq R^c$
- for all other congruence relations $R'$ with $R \subseteq R'$, either $R^c = R'$ or $R^c \subseteq R'$

$R^c$ is the smallest congruence relation that includes $R$.

**Example:** Let $S = \{a, b, c, d\}$ and $R = \{aRb, bRc, dRd\}$. Then

$$R^c = \{aRb, bRc, dRd, aRa, bRb, cRc, bRa, cRb, aRc, cRa\} \quad \text{since}$$

- $aRb, bRc, dRd \in R^c$ by $R \subseteq R^c$
- $aRa, bRb, cRc \in R^c$ by reflexivity
- $bRa, cRb \in R^c$ by symmetry
- $aRc \in R^c$ by transitivity
- $cRa \in R^c$ by symmetry
Deciding $\mathcal{T}_E$: Congruence Closure Algorithm

Consider the formula $F$:

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

where terms $s_i, t_i$ are terms. Is $F \mathcal{T}_E$-satisfiable?
Deciding $\mathcal{T}_E$: Congruence Closure Algorithm

Consider the formula $F$:

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

where terms $s_i, t_i$ are terms. Is $F \mathcal{T}_E$-satisfiable?

Idea

1. set $S$ is the set of subterms of $F$
2. construct the congruence class of each subterm of $F$ under the binary relation \{s_1 = t_1, \ldots, s_n = t_n\}. 
Deciding $\mathcal{T}_E$: Congruence Closure Algorithm

Consider the formula $F$:

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

where terms $s_i, t_i$ are terms. Is $F$ $\mathcal{T}_E$-satisfiable?

Idea

1. set $S$ is the set of subterms of $F$
2. construct the congruence class of each subterm of $F$ under the binary relation $\{s_1 = t_1, \ldots, s_n = t_n\}$.
3. if for some $i \in \{n + 1, \ldots, m\}$, we obtain that $s_i$ and $t_i$ are in the same congruence class, then $F$ is unsatisfiable. Otherwise, $F$ is satisfiable.
Deciding $\mathcal{T}_E$: Congruence Closure Algorithm

Consider the formula $F$:

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

where terms $s_i, t_i$ are terms. Is $F$ $\mathcal{T}_E$-satisfiable?

Idea

1. set $S$ is the set of subterms of $F$
2. construct the congruence class of each subterm of $F$ under the binary relation \{\(s_1 = t_1, \ldots, s_n = t_n\)\}.

**Congruence closure** $R$ of \{\(s_1 = t_1, \ldots, s_n = t_n\)\}!

3. if for some \(i \in \{n + 1, \ldots, m\}\), we obtain that $s_i$ and $t_i$ are in the same congruence class, then $F$ is unsatisfiable. Otherwise, $F$ is satisfiable.
Deciding $\mathcal{T}_E$: Congruence Closure Algorithm

procedure CongruenceClosure($F$)

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

output: satisfiable or unsatisfiable

parameters: function subterm_set

begin

$S_F := \text{subterm_set}(F)$

$R := \{sRs | s \in S_F\}$ and $[s]_R := \{s\}$, defining the partition $\{[s]_R | s \in S_F\}$

for every $s_i = t_i$ in $F$, merge $[s_i]_R$ and $[t_i]_R$ by

- forming the union $[s_i]_R \cup [t_i]_R$
- propagate the new congruences that arise in this union

if $s_j R t_j$ for any $j \in \{n + 1, \ldots, m\}$ then return unsatisfiable
else return satisfiable

end
Deciding $\mathcal{TE}$: Congruence Closure Algorithm

**procedure** CongruenceClosure($F$)

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

**output**: satisfiable or unsatisfiable

**parameters**: function `subterm_set`

**begin**

$S_F := \text{subterm\_set}(F)$

$R := \{sRs \mid s \in S_F\}$ and $[s]_R := \{s\}$, defining the partition $\{[s]_R \mid s \in S_F\}$

for every $s_i = t_i$ in $F$, merge $[s_i]_R$ and $[t_i]_R$ by

- forming the union $[s_i]_R \cup [t_i]_R$
- propagate the new congruences that arise in this union (function congruence)

if $s_j R t_j$ for any $j \in \{n+1, \ldots, m\}$ then **return** unsatisfiable

else **return** satisfiable **end**
Deciding $\mathcal{T}_E$: Congruence Closure by Example

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

Question: Is $F$ $\mathcal{T}_E$-satisfiable?
Deciding $\mathcal{T}_E$: Congruence Closure by Example

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

**Question:** Is $F$ $\mathcal{T}_E$-satisfiable?

Subterm set $\mathcal{S}_F = \{ a, b, f(a, b), f(f(a, b), b) \}$

Initial partition: $\{ \{ a \}, \{ b \}, \{ f(a, b) \}, \{ f(f(a, b), b) \} \}$

Using $f(a, b) = a$, merge $\{ f(a, b) \}$ and $\{ a \}$ and form partition:

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

From $f(a, b) R a$ and $b R b$, by congruence we have: $f(f(a, b), b) R f(a, b)$

Thus, merge $\{ a, f(a, b) \}$ and $\{ f(f(a, b), b) \}$ and form partition:

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

This is the partition of the congruence closure of $\{ f(a, b) = a \}$.

$F$ contains $f(f(a, b), b) \neq a$, but we have $f(f(a, b), b) R a$ in the congruence closure. Hence, $F$ is $\mathcal{T}_E$-unsatisfiable.
Deciding $\mathcal{T}_E$: Congruence Closure by Example

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

Question: Is $F$ $\mathcal{T}_E$-satisfiable?
Consider the formula \( F : f(a) = f(b) \land a \neq b. \)

**Question:** Is \( F \ \mathcal{T}_E \)-satisfiable?

Consider the formula
\[
F : f(f(f(a))) = a \land f(f(f(f(f(f(a))))) = a \land f(a) \neq a.
\]

**Question:** Is \( F \ \mathcal{T}_E \)-satisfiable?
Consider the formula $F : f(a, b) = a \land f(f(a, b), b) \neq a$.

A node is a subterm of $F$. 
Consider the formula $F : f(a, b) = a \land f(f(a, b), b) \neq a$. 

A node is a subterm of $F$. 

Initial congruence classes.
Consider the formula \( F : f(a, b) = a \land f(f(a, b), b) \neq a \).

A node is a subterm of \( F \).
Initial congruence classes.

A node is a subterm of \( F \).
Union of congruence classes.
Consider the formula $F : f(a, b) = a \land f(f(a, b), b) \neq a$. 

A node is a subterm of $F$. 
Initial congruence classes.

A node is a subterm of $F$. 
Union of congruence classes.

A node is a subterm of $F$. 
Propagation of congruence classes.
Consider the formula $F : f(a, b) = a \land f(f(a, b), b) \neq a$. 

A node is a subterm of $F$. Initial congruence classes. 

A node is a subterm of $F$. Union of congruence classes. 

A node is a subterm of $F$. Propagation of congruence classes.
Congruence Closure and DAGs

- A graph $G: \langle N, E \rangle$ has a set of nodes $N = \{n_1, \ldots, n_k\}$ and a set of edges $E = \{(n_i, n_j)\}_{i,j}$, where $n_i, n_j \in N$. 
Congruence Closure and DAGs

- A graph $G: \langle N, E \rangle$ has a set of nodes $N = \{n_1, \ldots, n_k\}$ and a set of edges $E = \{(n_i, n_j)\}_{i,j}$, where $n_i, n_j \in N$.

- A directed graph $G$ is a graph whose edges are directed from one node to another. For example, the edge $(n_1, n_2)$ is not the same as $(n_2, n_1)$.
A graph $G: \langle N, E \rangle$ has a set of nodes $N = \{n_1, \ldots, n_k\}$ and a set of edges $E = \{(n_i, n_j)\}_{i,j}$, where $n_i, n_j \in N$.

A directed graph $G$ is a graph whose edges are directed from one node to another. For example, the edge $(n_1, n_2)$ is not the same as $(n_2, n_1)$.

A directed acyclic graph (DAG) is a directed graph in which no subset of edges forms a directed loop/cycle.
Congruence Closure and DAGs

- A graph $G : \langle N, E \rangle$ has a set of nodes $N = \{n_1, \ldots, n_k\}$ and a set of edges $E = \{(n_i, n_j)\}_{i,j}$, where $n_i, n_j \in N$.
- A directed graph $G$ is a graph whose edges are directed from one node to another. For example, the edge $(n_1, n_2)$ is not the same as $(n_2, n_1)$.
- A directed acyclic graph (DAG) is a directed graph in which no subset of edges forms a directed loop/cycle.

Congruence closure with DAG for a formula $F$ in $\mathcal{T}_E$:

- A DAG node represents a subterm of $F$;
- Congruence classes are stored via references between DAG nodes.
Consider the formula $F : f(a, b) = a \land f(f(a, b), b) \neq a$.

The DAG of its subterms:
Congruence Closure and DAGs

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

The DAG of its subterms:

A node $n$ has an:

- unique number identifier $id$;
- root function/constant symbol $fn$ of the subterm represented by $n$;
- list $args$ of identifiers of the nodes representing the function arguments of $n$;
Consider the formula $F : f(a, b) = a \land f(f(a, b), b) \neq a$.

The DAG of its subterms:

A node $n$ has an:

- unique number identifier $\text{id}$;
- root function/constant symbol $\text{fn}$ of the subterm represented by $n$;
- list $\text{args}$ of identifiers of the nodes representing the function arguments of $n$;

Node representing $f(a, b)$ has: $\text{id} = 2, \text{fn} = f, \text{args} = \{3, 4\}$
Congruence Closure and DAGs

The DAG of \( F : f(a, b) = a \land f(f(a, b), b) \neq a \) after the union of congruence classes for \( f(a, b) = a \):

A node \( n \) has an:

- unique number identifier id;
- root function/constant symbol fn of the subterm represented by \( n \);
- list args of identifiers of the nodes representing the function arguments of \( n \);
- identifier find of another node (possibly itself) in the congruence class of \( n \). A representative of a congruence has itself as find;
Congruence Closure and DAGs

The DAG of $F : f(a, b) = a \land f(f(a, b), b) \neq a$ after the union of congruence classes for $f(a, b) = a$:

A node $n$ has an:

- unique number identifier $id$;
- root function/constant symbol $fn$ of the subterm represented by $n$;
- list $args$ of identifiers of the nodes representing the function arguments of $n$;
- identifier $find$ of another node (possibly itself) in the congruence class of $n$. A representative of a congruence has itself as $find$;

Node representing $f(a, b)$ has: $find = 3$
Node representing $b$ has: $find = 4$
Congruence Closure and DAGs

The DAG of \( F : f(a, b) = a \land f(f(a, b), b) \neq a \) after the propagation of congruence classes for \( f(a, b) = a \):

A node \( n \) has an:

- unique number identifier id;
- root function/constant symbol fn of the subterm represented by \( n \);
- list args of identifiers of the nodes representing the function arguments of \( n \);
- identifier find of another node (possibly itself) in the congruence class of \( n \); A representative of a congruence has itself as find;
- list ccpar of identifiers of all parents of all nodes in the congruence class of \( n \), if \( n \) is a representative of its congruence class. Otherwise, ccpar is empty.
The DAG of $F : f(a, b) = a \land f(f(a, b), b) \neq a$ after the propagation of congruence classes for $f(a, b) = a$:

1. Node representing $f(f(a, b), b)$ has: $\text{find} = 3$ and $\text{ccpar} = \emptyset$;
2. Node representing $f(a, b)$ has: $\text{ccpar} = \emptyset$;
3. Node representing $b$ has: $\text{ccpar} = \{1, 2\}$;
4. Node representing $a$: $\text{ccpar} = \{1, 2\}$;
Congruence Closure and DAGs with Union-Find

- Union-Find algorithm on the DAG for congruence closure;
- Find: for computing the representative of a congruence class;
- Union: for taking the union of two congruence classes;
Congruence Closure and DAGs with Union-Find

- Union-Find algorithm on the DAG for congruence closure;
- Find: for computing the representative of a congruence class;
- Union: for taking the union of two congruence classes;
- Merge: for merging two congruence classes, by taking their union and propagating new congruences (function congruence);
Congruence Closure and DAGs with Union-Find

**procedure** `Node(i)`

**input**: a DAG and an identifier `i`

**output**: Node `n` with id `i`

Example for the DAG of $F : f(a, b) = a \land f(f(a, b), b) \neq a$: `Node(3).id = 3`
**Congruence Closure and DAGs with Union-Find**

**procedure** $Node(i)$
**input:** a DAG and an identifier $i$
**output:** Node $n$ with id $i$

**procedure** $Find(i)$
**input:** a DAG and an identifier $i$
**output:** representative of the congruence class of the node with id $i$

```
begin
    n := $Node(i)$
    if $n.find = i$ then return $i$
    else return $Find(n.find)$
end
```
Congruence Closure and DAGs with Union-Find

procedure $Node(i)$
input: a DAG and an identifier $i$
output: Node $n$ with id $i$

procedure $Find(i)$
input: a DAG and an identifier $i$
output: representative of the congruence class of the node with id $i$
begin
    $n := Node(i)$
    if $n.find = i$ then return $i$
    else return $Find(n.find)$
end

Example for $F : f(a, b) = a \land f(f(a, b), b) \neq a$:
In the initial DAG of $F$: $Find(2) = 2$.

In the DAG of $F$ after the union of congruence classes for $f(a, b) = a$: $Find(2) = 3$
Congruence Closure and DAGs with Union-Find

procedure Union($i_1, i_2$)

input: a DAG and identifiers $i_1, i_2$

output: modified DAG with the union of the equivalence classes of the nodes with id $i_1$ and $i_2$, with updated ccpar and find for these nodes

begin

$n_1 := \text{Node}(\text{Find}(i_1))$

$n_2 := \text{Node}(\text{Find}(i_2))$

$n_1.\text{find} := n_2.\text{find}$

$n_2.\text{ccpar} := n_1.\text{ccpar} \cup n_2.\text{ccpar}$

$n_1.\text{ccpar} = \emptyset$

end
procedure \textit{Union}(i_1, i_2)
\textbf{input}: a DAG and identifiers \(i_1\), \(i_2\)
\textbf{output}: modified DAG with the union of the equivalence classes of the nodes
with id \(i_1\) and \(i_2\), with updated \texttt{ccpar} and \texttt{find} for these nodes
\begin{verbatim}
begin
\quad n_1 := Node(Find(i_1))
\quad n_2 := Node(Find(i_2))
\quad n_1.\texttt{find} := n_2.\texttt{find}
\quad n_2.\texttt{ccpar} := n_1.\texttt{ccpar} \cup n_2.\texttt{ccpar}
\quad n_1.\texttt{ccpar} = \emptyset
end
\end{verbatim}

In the DAG of \(F: f(a, b) = a \land f(f(a, b), b) \neq a\) after the union of congruence
classes for \(f(a, b) = a\):

\textbf{Union}(1, 2) sets:
1.\texttt{find} = \textit{Node}(\textit{Find}(2)).\texttt{find} = 3.\texttt{find} = 3;
3.\texttt{ccpar} = \{1, 2\} and 1.\texttt{ccpar} = \emptyset.
Congruence Closure and DAGs with Union-Find

**procedure** `Union(i_1, i_2)`

**input:** a DAG and identifiers $i_1$, $i_2$

**output:** modified DAG with the union of the equivalence classes of the nodes with id $i_1$ and $i_2$, with updated ccpar and find for these nodes

**begin**

$n_1 := Node(Find(i_1))$

$n_2 := Node(Find(i_2))$

$n_1$.find := $n_2$.find

$n_2$.ccpar := $n_1$.ccpar $\cup$ $n_2$.ccpar

$n_1$.ccpar := $\emptyset$

**end**

**procedure** `CCPAR(i)`

**input:** a DAG and an identifier $i$

**output:** the parents of all nodes in the congruence class of the node with id $i$

**begin**

return `Node(Find(i)).ccpar`

**end**
procedure Congruent\(i_1, i_2\)

\textbf{input:} a DAG and identifiers \(i_1, i_2\)

\textbf{output:} \textit{True} if the nodes with id \(i_1\) and \(i_2\) are congruent, otherwise \textit{False}

\begin{verbatim}
begin
    n_1 := Node(i_1)
    n_2 := Node(i_2)
    if n_1.fn = n_2.fn and |n_1.args| = |n_2.args| and
        for all \(i \in \{1, \ldots, |n_1.args|\} : Find(n_1.args[i]) = Find(n_2.args[i])\)
    then return \textit{True}
end
\end{verbatim}
**procedure** *Congruent*(*i₁*, *i₂*)

**input**: a DAG and identifiers *i₁*, *i₂*

**output**: *True* if the nodes with id *i₁* and *i₂* are congruent, otherwise *False*

**begin**

\[ n₁ := \text{Node}(i₁) \]
\[ n₂ := \text{Node}(i₂) \]

\[ \text{if } n₁.\text{fn} = n₂.\text{fn} \text{ and } |n₁.\text{args}| = |n₂.\text{args}| \text{ and } \]

\[ \text{for all } i \in \{1, \ldots, |n₁.\text{args}|\} : \text{Find}(n₁.\text{args}[i]) = \text{Find}(n₂.\text{args}[i]) \]

**then** return *True*

**end**

In the DAG of \( F : f(a, b) = a \land f(f(a, b), b) \neq a \) after the union of congruence classes for \( f(a, b) = a \):

Are nodes with id 1 and 2 congruent? Why?
procedure \textit{Merge}(i_1, i_2)
\textbf{input}: a DAG and identifiers \(i_1, i_2\)
\textbf{output}: modified DAG with the merged congruence classes of the nodes
with id \(i_1\) and \(i_2\) (union and propagation)
\begin{algorithmic}
\State \textbf{if} \textit{Find}(i_1) = \textit{Find}(i_2) \textbf{then} \textbf{return}
\textbf{else}
\State \(P_{i_1} := \text{CCPAR}(i_1)\)
\State \(P_{i_2} := \text{CCPAR}(i_2)\)
\State \textit{Union}(i_1, i_2)
\State \textbf{for each} \((t_1, t_2) \in P_{i_1} \times P_{i_2}\) \textbf{do}
\State \textbf{if} \textit{Find}(t_1) \neq \textit{Find}(t_2) \textbf{and} \textit{Congruent}(t_1, t_2) \textbf{then} \textit{Merge}(t_1, t_2)
\State \textbf{end do}
\textbf{end}
\end{algorithmic}
Deciding $\mathcal{T}_E$: Congruence Closure with DAGs

procedure $CongruenceClosure_{\_wDAG}(F)$
input: $F$ is $s_1 = t_1 \land \ldots s_n = t_n \land s_{n+1} \neq t_{n+1} \land \ldots s_m \neq t_m$
output: satisfiable or unsatisfiable
parameters: procedure $Merge$
begin
  Construct the initial DAG for the subterm set $S_F$ of $F$
  for each $i \in \{1, \ldots, n\}$ do
    $Merge(s_i, t_i)$
  end do
  if $Find(s_i) = Find(t_i)$ for some $i \in \{n+1, \ldots m\}$ return unsatisfiable
  else return satisfiable
end
Deciding $\mathcal{T}_E$: Congruence Closure with DAGs

**procedure** CongruenceClosure$_{wDAG}(F)$

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

**output:** *satisfiable* or *unsatisfiable*

**parameters:** procedure *Merge*

**begin**

Construct the initial DAG for the subterm set $S_F$ of $F$

for each $i \in \{1, \ldots, n\}$ do

$\text{Merge}(s_i, t_i)$

end do

if $\text{Find}(s_i) = \text{Find}(t_i)$ for some $i \in \{n + 1, \ldots m\}$ return *unsatisfiable*

else return *satisfiable*

end

Is $f(a, b) = a \land f(f(a, b), b) \neq a$ $\mathcal{T}_E$-satisfiable?
Deciding $\mathcal{T}_E$: Congruence Closure with DAGs

```
procedure CongruenceClosure_wDAG(F)
input: $F$ is $s_1 = t_1 \land \ldots s_n = t_n \land s_{n+1} \neq t_{n+1} \land \ldots s_m \neq t_m$
output: satisfiable or unsatisfiable
parameters: procedure Merge
begin
    Construct the initial DAG for the subterm set $S_F$ of $F$
    for each $i \in \{1, \ldots, n\}$ do
        Merge($s_i, t_i$)
    end do
    if Find($s_i$) = Find($t_i$) for some $i \in \{n + 1, \ldots, m\}$ return unsatisfiable
    else return satisfiable
end
```

The $\text{CongruenceClosure}_w\text{DAG}(F)$ algorithms runs in time $O(e^2)$ for $O(n)$ merges, where $e$ is the number of edges and $n$ the number of nodes in the initial DAG of $F$.

Computing $\mathcal{T}_E$-satisfiability is inexpensive.