Software Verification with Satisfiability Modulo Theories
- Decision Procedures -

Ming-Hsien Tsai

Institute of Information Science
Academia Sinica

FLOLAC 2017

Reference book: Aaron R. Bradley and Zohar Manna. The Calculus of Computation. Springer 2007

Outline

- The theory T_E and its quantifier-free fragment
- Deciding T_E-satisfiability of quantifier-free Σ_E-formulae
  - Congruence closure algorithm
- Implementation of the decision procedure
- T_{RDS} - recursive data structures
  - T_{cons} - lists
- T_A - arrays

Theory of Equality

Theory of Equality

- Denoted by T_E
- Referred to as the theory of EUF (Equality with Uninterpreted Functions)
- Play a central role in combining theories that share the equality predicate

Signature of T_E

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

consists of

- $=$, a binary predicate;
- and all constant, function and predicate symbols

Σ_E-formulae

- $x = g(y, x) \rightarrow f(x) = f(g(y, z))$
- $f(f(f(a))) = a \wedge f(f(f(f(f(a))))) = a \wedge f(a) \neq a$

$f(a) \neq a$ abbreviates $\neg(f(a) = a)$

Axioms of Equality

- Reflexivity: $\forall x.\ x = x$
- Symmetry: $\forall x, y.\ x = y \rightarrow y = x$
- Transitivity: $\forall x, y, z.\ x = y \wedge y = z \rightarrow x = z$

Axioms of Equality

- Reflexivity: $\forall x.\ x = x$
- Symmetry: $\forall x,y.\ x = y \to y = x$
- Transitivity: $\forall x,y,z.\ x = y \land y = z \to x = z$

with the three axioms, $=$ is defined
to be an equivalence relation

Equality of Function Terms

- When two function terms are equal?

$$f(x) = f(g(y,\, z))$$

Function Congruence

- Function congruence (axiom schema)
  - $\forall X, Y.\ (\wedge_{i=1 \text{ to } n}\ x_i = y_i) \rightarrow f(X) = f(Y)$
- Instantiated axioms:
  - $\forall x, y.\ x = y \rightarrow f(x) = f(y)$
  - $\forall x_1, x_2, y_1, y_2.\ x_1 = y_1 \wedge x_2 = y_2 \rightarrow g(x_1,\ x_2) = g(y_1,\ y_2)$

Capital X and Y are vectors of variables

Function Congruence

- Function congruence (axiom schema)
  - $\forall X, Y.\ (\wedge_{i=1 \text{ to } n}\ x_i = y_i) \rightarrow f(X) = f(Y)$
- Instantiated axioms:
  - $\forall x, y.\ x = y \rightarrow f(x) = f(y)$
  - $\forall x_1, x_2, y_1, y_2.\ x_1 = y_1 \wedge x_2 = y_2 \rightarrow g(x_1, x_2) = g(y_1, y_2)$

makes $=$ a congruence relation

Capital X and Y are vectors of variables

Predicate Congruence

- Predicate congruence
  - $\forall X, Y.\ (\wedge_{i=1 \text{ to } n}\ x_i = y_i) \rightarrow (p(X) \leftrightarrow p(Y))$

T_E-Satisfiablility - Example 1

- Is the following Σ_E-formula T_E-satisfiable?
  - $f(x) = f(y) \land x \neq y$

$x \neq y$ abbreviates $\neg(x = y)$

T_E-Satisfiablility - Example 2

Is the following Σ_E-formula T_E-satisfiable?

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

T_E-Satisfiablility - Example 2

Is the following Σ_E-formula T_E-satisfiable?

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

1. $f(f(f(f(a)))) = f(a)$ **(function congruence)**

T_E-Satisfiablility - Example 2

Is the following Σ_E-formula T_E-satisfiable?

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

1. $f(f(f(f(a)))) = f(a)$ **(function congruence)**
2. $f(f(f(f(f(a))))) = f(f(a))$ **(function congruence)**

T_E-Satisfiablility - Example 2

Is the following Σ_E-formula T_E-satisfiable?

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

1. $f(f(f(f(a)))) = f(a)$ **(function congruence)**

2. $f(f(f(f(f(a))))) = f(f(a))$ **(function congruence)**

3. $f(f(a)) = f(f(f(f(f(a)))))$ **(symmetry)**

T_E-Satisfiablility - Example 2

Is the following Σ_E-formula T_E-satisfiable?

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

1. $f(f(f(f(a)))) = f(a)$ **(function congruence)**

2. $f(f(f(f(f(a))))) = f(f(a))$ **(function congruence)**

3. $f(f(a)) = f(f(f(f(f(a)))))$ **(symmetry)**

4. $f(f(a)) = a$ **(transitivity)**

Get Rid of Predicate Congruence

- Transform a Σ_E-formula to a Σ_E-formula without predicates other than $=$
- Example p1
  - $x = y \to (p(x) \leftrightarrow p(y))$ is transformed to
  - $x = y \to ((f_p(x) = \bullet) \leftrightarrow (f_p(y) = \bullet))$
- Example p2
  - $p(x) \wedge q(x, y) \wedge q(y, z) \to \neg q(x, z)$ is transformed to
  - $f_p(x) = \bullet \wedge f_q(x, y) = \bullet \wedge f_q(y, z) = \bullet \to f_q(x, z) \neq \bullet$

In The Following

- Consider Σ_E-formulae without predicates other than $=$
- T_E-satisfiability of Σ_E-formulae is undecidable
  - Consider only the quantifier-free fragment
  - Consider formulae in disjunctive normal form (DNF)

$$(a_1 \wedge a_2 \wedge \ldots \wedge a_n) \vee \ldots \vee (b_1 \wedge b_2 \wedge \ldots \wedge b_m)$$

Congruence Closure Algorithm

Observation

- Applying (symmetry), (reflexivity), (transitivity), and (congruence) to positive literals $s = t$ of a Σ_E-formula F produces more equalities over terms occurring in formula F

- There are only a finite number of terms in F

- Only a finite number of equalities among these terms are possible

- Then, either

  - some equality is formed that directly contradicts a negative literal $s' \neq t'$ of F; or

  - the propagation of equalities ends without finding a contradiction

Observation

- Applying (symmetry), (reflexivity), (transitivity), and (congruence) to positive literals $s = t$ of a Σ_E-formula F produces more equalities over terms occurring in formula F

- There are only a finite number of terms in F

- Only a finite number of equalities among these terms are possible

- Then, either

  - some equality is formed that directly contradicts a negative literal $s' \neq t'$ of F; or

  - the propagation of equalities ends without finding a contradiction

form the congruence closure of $=$

Class

- Consider an equivalence relation R over a set S
- 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 over S, then $[s]_R$ is the *congruence class* of s

Example of Class

- Consider the set $\mathbb{Z}$ of integers and the equivalence relation $\equiv_2$ such that
  - $m \equiv_2 n$ iff $(m\ mod\ 2) = (n\ mod\ 2)$

$$
\begin{aligned}
[3]_{\equiv_2} &= \{n \in \mathbb{Z} : (n\ mod\ 2) = (3\ mod\ 2)\} \\
&= \{n \in \mathbb{Z} : (n\ mod\ 2) = 1\} \\
&= \{n \in \mathbb{Z} : n \text{ is odd}\}
\end{aligned}
$$

Partition

A *partition* P of S is a set of subsets of S that is *total*,

$$\left(\cup_{S' \in P} S'\right) = S,$$

and *disjoint*,

$$\forall S_1, S_2 \in P.\ S_1 \neq S_2 \rightarrow S_1 \cap S_2 = \varnothing$$

Quotient

- The *quotient* S/R of S by the equivalence (congruence) relation R is a partition of S: it is a set of equivalence (congruence) classes

- $S/R = \{[s]_R : s \in S\}$

Example of Quotient

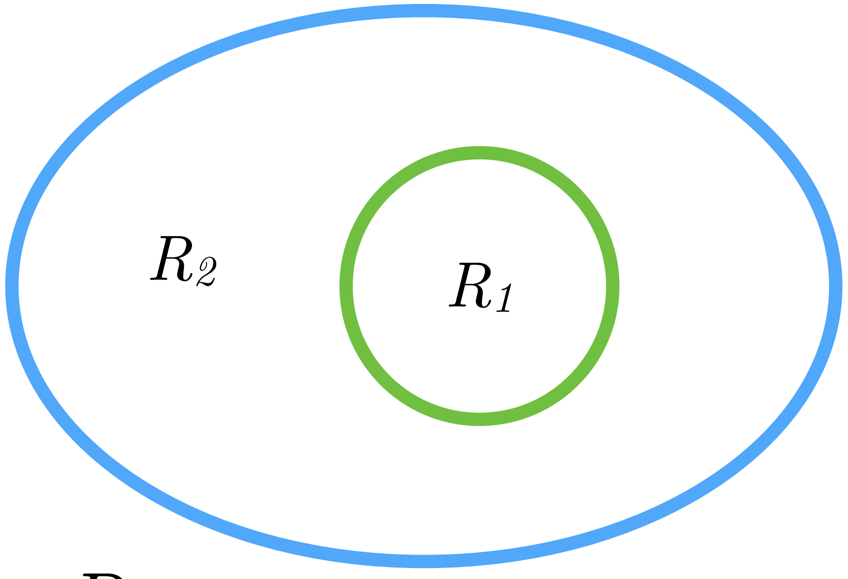
- The quotient $\mathbb{Z}/{\equiv_2}$ is a partition: it is the set of equivalence classes
  - $\{\{n \in \mathbb{Z} : n \text{ is odd}\}, \{n \in \mathbb{Z} : n \text{ is even}\}\}$

Equivalence Relation, Partition, and Quotient

- An equivalence relation R induces a partition S/R of S
- A given partition P of S induces an equivalence relation over S
  - $s_1 R s_2$ iff for some $S' \in P$, both $s_1, s_2 \in S'$

Relation Refinement

- Consider two binary relations R_1 and R_2 over the set S
- R_1 is a *refinement* of R_2, or $R_1 < R_2$, if
  - $\forall s_1, s_2 \in S.\ s_1 R_1 s_2 \to s_1 R_2 s_2$
- We also say that R_1 refines R_2
- Viewing the relations as sets of pairs, $R_1 \subseteq R_2$



Example 1 of Relation Refinement

- $S = \{a,\ b\}$
- $R_1 : \{aR_1b\}$
- $R_2 : \{aR_2b,\ bR_2b\}$
- $R_1 < R_2$

Example 2 of Relation Refinement

- Consider set S
- $R_1 : \{sR_1s : s \in S\}$
- $R_2 : \{sR_2t : s,t \in S\ \}$
- $R_1 < R_2$

Example 2 of Relation Refinement

- Consider set S
- $R_1 : \{sR_1s : s \in S\}$ $P_1 : \{\{s\} : s \in S\}$
- $R_2 : \{sR_2t : s,t \in S \}$
- $R_1 < R_2$

Example 2 of Relation Refinement

- Consider set S
- $R_1 : \{sR_1s : s \in S\}$ $P_1 : \{\{s\} : s \in S\}$
- $R_2 : \{sR_2t : s,t \in S\}$ $P_2 : \{S\}$
- $R_1 < R_2$

Example 3 of Relation Refinement

- Consider the set $\mathbb{Z}$
- $R_1 : \{xR_1y : x\ mod\ 2 = y\ mod\ 2\}$
- $R_2 : \{xR_1y : x\ mod\ 4 = y\ mod\ 4\}$
- $R_2 < R_1$

Closure

- The *equivalence closure* R^E of the binary relation R over S is the equivalence relation such that
  - R refines R^E: $R \prec R^E$;
  - for all other equivalence relations R' such that $R \prec R$', either
    - $R' = R^E$, or
    - $R^E \prec R$'
- R^E is the smallest equivalence relation that covers R

Example of Equivalence Closure

- Then,
  - $aRb,\ bRc,\ dRd \in R^E$ (since $R \subseteq R^E$);
  - $aRa,\ bRb,\ cRc \in R^E$ (by reflexivity);
  - $bRa,\ cRb \in R^E$ (by symmetry);
  - $aRc \in R^E$ (by transitivity);
  - $cRa \in R^E$ (by symmetry);

$$S = \{a,\ b,\ c,\ d\}$$

$$R = \{aRb,\ bRc,\ dRd\}$$

- Hence
  - $R^E = \{aRb,\ bRa,\ aRa,\ bRb,\ bRc,\ cRb,\ cRc,\ aRc,\ cRa,\ dRd\}$

Congruence Closure

- The *congruence closure* R^C of R is the smallest congruence relation that covers R

Congruence Closure

- The *congruence closure* R^C of R is the smallest congruence relation that covers R

Compute the congruence closure of a term set

Subterm Set

- *Subterm set* S_F of Σ_E-formula F is the set that contains precisely the subterms of F

- Example:

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

  - $S_F = \{a, b, f(a, b), f(f(a, b), b)\}$

Congruence Relation over Subterm Set

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

- F is T_E-satisfiable iff there exists a congruence relation $\sim$ over S_F such that
  - for each $i \in \{1, \ldots, m\}$, $s_i \sim t_i$;
  - for each $i \in \{m+1, \ldots, n\}$, $s_i \nsim t_i$

T_E-interpretation

- The congruence relation $\sim$ defines a T_E-interpretation $I : (D_{\mathrm{I}}, \alpha_I)$ of F
  - D_I consists of $|S_F \;/\; \sim|$ elements
  - α_I assigns elements of D_I to the terms of S_F in a way that respects $\sim$
  - α_I assigns to $=$ a binary relation over D_I that behaves like $\sim$
- We abbreviate $(D_I,\ \alpha_I) \vDash F$ with $\sim\, \vDash F$

Congruence Closure Algorithm

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

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 $i \in \{m+1, \ldots, n\}$, return unsatisfiable

3. Otherwise, $\sim \models F$, so return satisfiable

Step 1

- Begin with $\sim_0$ given by the partition $\{\{s\} : s \in S_F\}$
- Import $s_i = t_i$ by merging the congruence classes $[s_i]_{\sim i\text{-}1}$ and $[t_i]_{\sim i\text{-}1}$
  - Form the union of $[s_i]_{\sim i\text{-}1}$ and $[t_i]_{\sim i\text{-}1}$
  - Propagate new congruences that arise within the union

Example 1 of Congruence Closure Algorithm

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

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

Example 1 of Congruence Closure Algorithm

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

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

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

Example 1 of Congruence Closure Algorithm

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

- $\{\{a\},\ \{b\},\ \{f(a,\ b)\},\ \{f(f(a,\ b),\ b)\}\}$
- $\{\{a,\ \mathrm{f}(a,\ b)\},\ \{b\},\ \{f(f(a,\ b),\ b)\}\}$ $\quad (f(a,\ b) = a)$

Example 1 of Congruence Closure Algorithm

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

- $\{\{a\},\ \{b\},\ \{f(a,\ b)\},\ \{f(f(a,\ b),\ b)\}\}$
- $\{\{a,\ \mathrm{f}(a,\ b)\},\ \{b\},\ \{f(f(a,\ b),\ b)\}\}$ $(f(a,\ b) = a)$

(function congruence)

Example 1 of Congruence Closure Algorithm

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

- $\{\{a\},\ \{b\},\ \{f(a,\ b)\},\ \{f(f(a,\ b),\ b)\}\}$
- $\{\{a,\ \mathrm{f}(a,\ b)\},\ \{b\},\ \{f(f(a,\ b),\ b)\}\}$ $(f(a,\ b) = a)$
- $\{\{a,\ f(a,\ b),\ f(f(a,\ b),\ b)\},\ \{b\}\}$ (function congruence)

Example 1 of Congruence Closure Algorithm

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

- $\{\{a\},\ \{b\},\ \{f(a,\ b)\},\ \{f(f(a,\ b),\ b)\}\}$
- $\{\{a,\ \mathrm{f}(a,\ b)\},\ \{b\},\ \{f(f(a,\ b),\ b)\}\}$ $(f(a,\ b) = a)$
- $\{\{a,\ f(a,\ b),\ f(f(a,\ b),\ b)\},\ \{b\}\}$ (function congruence)
- T_E-unsatisfiable

Example 2 of Congruence Closure Algorithm

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

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

Example 2 of Congruence Closure Algorithm

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

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

$(f^3(a) = a)$

Example 2 of Congruence Closure Algorithm

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

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

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

Example 2 of Congruence Closure Algorithm

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

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

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

(function congruence)

Example 2 of Congruence Closure Algorithm

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

- $\{\{a\}, \{f(a)\}, \{f^2(a)\}, \{f^3(a)\}, \{f^4(a)\}, \{f^5(a)\}\}$
- $\{\{a, f^3(a)\}, \{f(a)\}, \{f^2(a)\}, \{f^4(a)\}, \{f^5(a)\}\}$ $(f^3(a) = a)$
- $\{\{a, f^3(a)\}, \{f(a), f^4(a)\}, \{f^2(a), f^5(a)\}\}$ (function congruence)

Example 2 of Congruence Closure Algorithm

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

- $\{\{a\}, \{f(a)\}, \{f^2(a)\}, \{f^3(a)\}, \{f^4(a)\}, \{f^5(a)\}\}$
- $\{\{a, f^3(a)\}, \{f(a)\}, \{f^2(a)\}, \{f^4(a)\}, \{f^5(a)\}\}$ $(f^3(a) = a)$
- $\{\{a, f^3(a)\}, \{f(a), f^4(a)\}, \{f^2(a), f^5(a)\}\}$ (function congruence)

$(f^5(a) = a)$

Example 2 of Congruence Closure Algorithm

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

- $\{\{a\}, \{f(a)\}, \{f^2(a)\}, \{f^3(a)\}, \{f^4(a)\}, \{f^5(a)\}\}$
- $\{\{a, f^3(a)\}, \{f(a)\}, \{f^2(a)\}, \{f^4(a)\}, \{f^5(a)\}\}$ $(f^3(a) = a)$
- $\{\{a, f^3(a)\}, \{f(a), f^4(a)\}, \{f^2(a), f^5(a)\}\}$ (function congruence)
- $\{\{a, f^2(a), f^3(a), f^5(a)\}, \{f(a), f^4(a)\}\}$ $(f^5(a) = a)$

Example 2 of Congruence Closure Algorithm

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

- $\{\{a\}, \{f(a)\}, \{f^2(a)\}, \{f^3(a)\}, \{f^4(a)\}, \{f^5(a)\}\}$
- $\{\{a, f^3(a)\}, \{f(a)\}, \{f^2(a)\}, \{f^4(a)\}, \{f^5(a)\}\}$ $(f^3(a) = a)$
- $\{\{a, f^3(a)\}, \{f(a), f^4(a)\}, \{f^2(a), f^5(a)\}\}$ (function congruence)
- $\{\{a, f^2(a), f^3(a), f^5(a)\}, \{f(a), f^4(a)\}\}$ $(f^5(a) = a)$

(function congruence)

Example 2 of Congruence Closure Algorithm

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

- $\{\{a\}, \{f(a)\}, \{f^2(a)\}, \{f^3(a)\}, \{f^4(a)\}, \{f^5(a)\}\}$
- $\{\{a, f^3(a)\}, \{f(a)\}, \{f^2(a)\}, \{f^4(a)\}, \{f^5(a)\}\}$ $(f^3(a) = a)$
- $\{\{a, f^3(a)\}, \{f(a), f^4(a)\}, \{f^2(a), f^5(a)\}\}$ (function congruence)
- $\{\{a, f^2(a), f^3(a), f^5(a)\}, \{f(a), f^4(a)\}\}$ $(f^5(a) = a)$
- $\{\{a, f(a), f^2(a), f^3(a), f^4(a), f^5(a)\}\}$ (function congruence)

Example 2 of Congruence Closure Algorithm

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

- $\{\{a\}, \{f(a)\}, \{f^2(a)\}, \{f^3(a)\}, \{f^4(a)\}, \{f^5(a)\}\}$
- $\{\{a, f^3(a)\}, \{f(a)\}, \{f^2(a)\}, \{f^4(a)\}, \{f^5(a)\}\}$ $(f^3(a) = a)$
- $\{\{a, f^3(a)\}, \{f(a), f^4(a)\}, \{f^2(a), f^5(a)\}\}$ (function congruence)
- $\{\{a, f^2(a), f^3(a), f^5(a)\}, \{f(a), f^4(a)\}\}$ $(f^5(a) = a)$
- $\{\{a, f(a), f^2(a), f^3(a), f^4(a), f^5(a)\}\}$ (function congruence)

T_E-unsatisfiable

Example 3 of Congruence Closure Algorithm

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

- $\{\{x\}, \{y\}, \{f(x)\}, \{f(y)\}\}$

Example 3 of Congruence Closure Algorithm

$$F : f(x) = f(y) \wedge x \neq y$$

- $\{\{x\}, \{y\}, \{f(x)\}, \{f(y)\}\}$

$$(f(x) = f(y))$$

Example 3 of Congruence Closure Algorithm

$$F : f(x) = f(y) \wedge x \neq y$$

- $\{\{x\}, \{y\}, \{f(x)\}, \{f(y)\}\}$
- $\{\{x\}, \{y\}, \{f(x), f(y)\}\}$ $\quad (f(x) = f(y))$

Example 3 of Congruence Closure Algorithm

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

- $\{\{x\}, \{y\}, \{f(x)\}, \{f(y)\}\}$
- $\{\{x\}, \{y\}, \{f(x), f(y)\}\}$ $\quad (f(x) = f(y))$
- T_E-satisfiable

Exercise

- Apply the decision procedure for T_E to the following Σ_E-formulas. Provide a level of details as in the slides.

1. $f(x,y) = f(y,x) \wedge f(a,y) \neq f(y,a)$

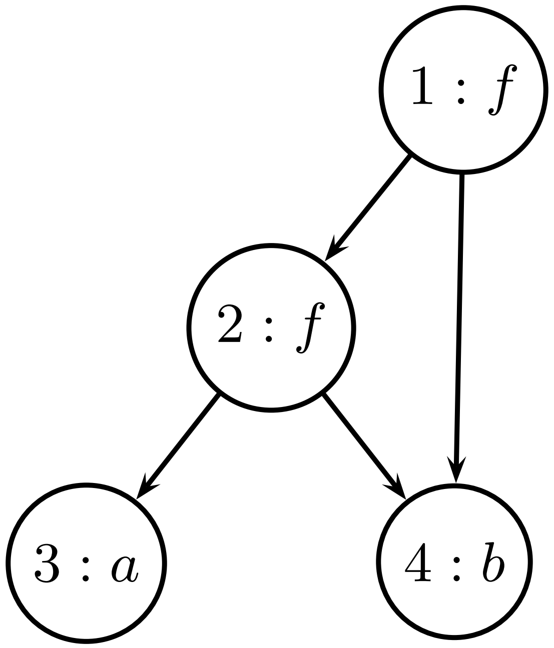
2. $f(g(x)) = g(f(x)) \wedge f(g(f(y))) = x \wedge f(y) = x \wedge g(f(x)) \neq x$

3. $f(f(f(a))) = f(f(a)) \wedge f(f(f(f(a)))) = a \wedge f(a) \neq a$

4. $p(x) \wedge f(f(x)) = x \wedge f(f(f(x))) = x \wedge \neg p(f(x))$

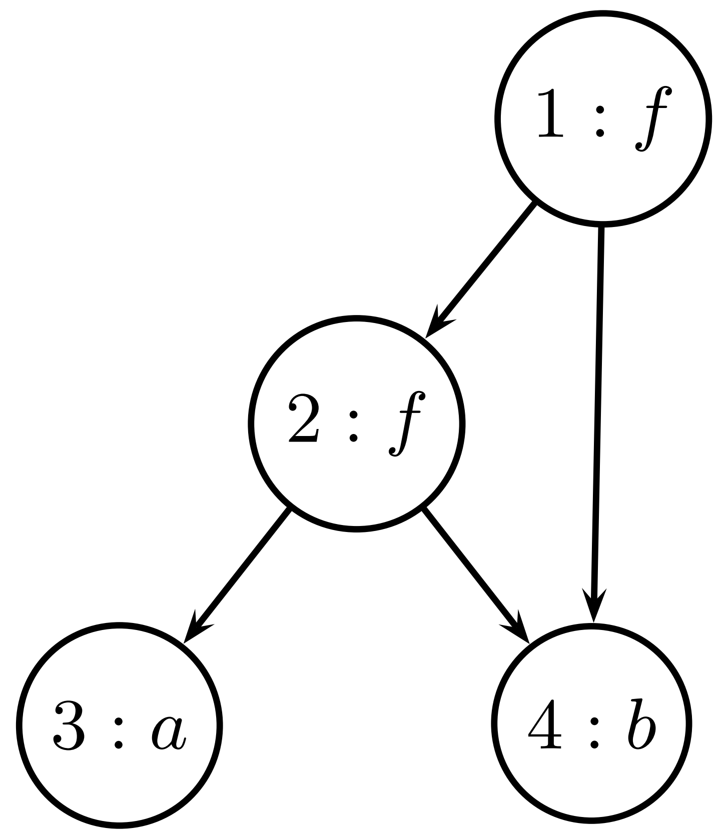
Implementation

DAG



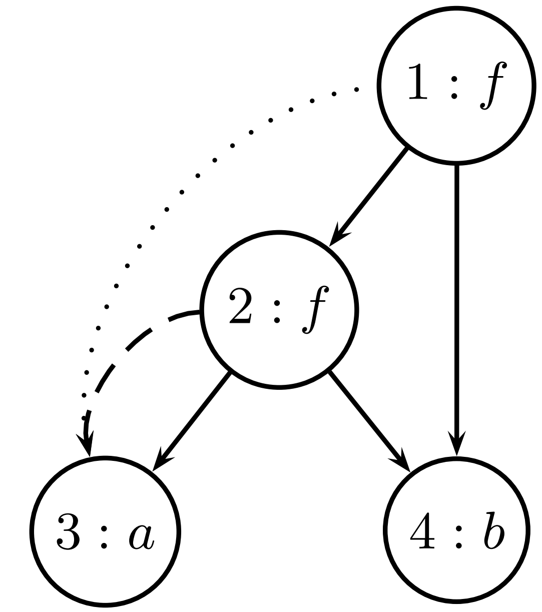
- A directed graph $G : \langle N,\ E \rangle$
  - nodes $N = \{n_1,\ n_2,\ \ldots,\ n_k\}$
  - edges $E = \{\ldots,\ \langle n_i,\ n_j \rangle,\ \ldots\}$
- A *directed acyclic graph* (*DAG*) is a directed graph containing no loop (or cycle)

Subterm Set as DAG



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

Node



type node = {

id : id — (unique identification number)

fn : string — (constant or function symbol)

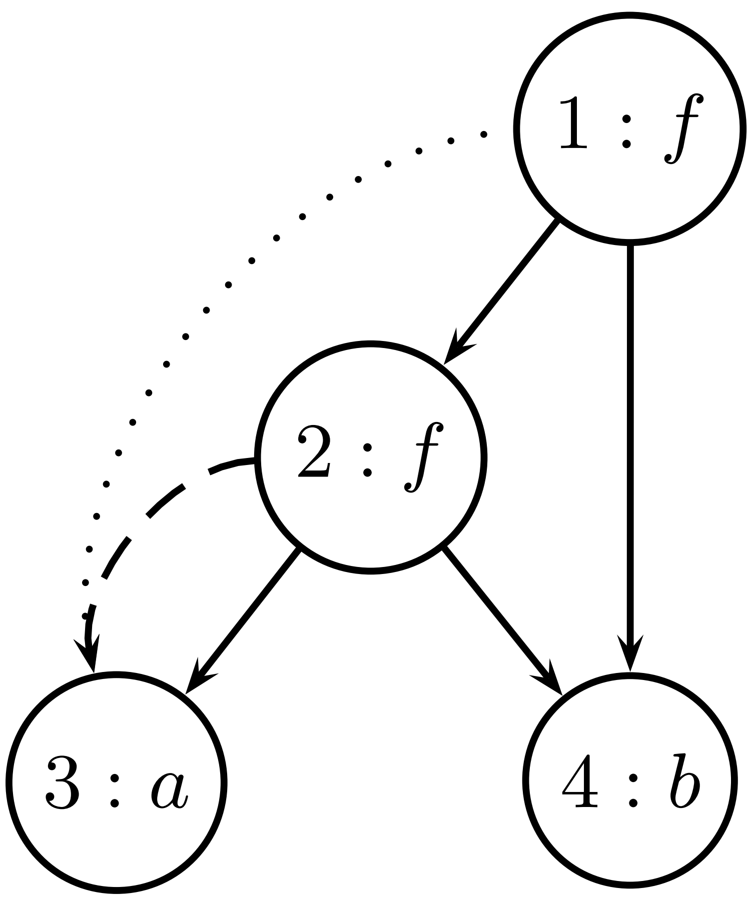
args : id list — (identification numbers of the function arguments)

mutable find : id — (another node in its congruence class)
(following a chain of find references leads to the representative)

mutable ccpar : id set — (congruence closure parents, ∅ for non-representative nodes)

}

DAG as Partition



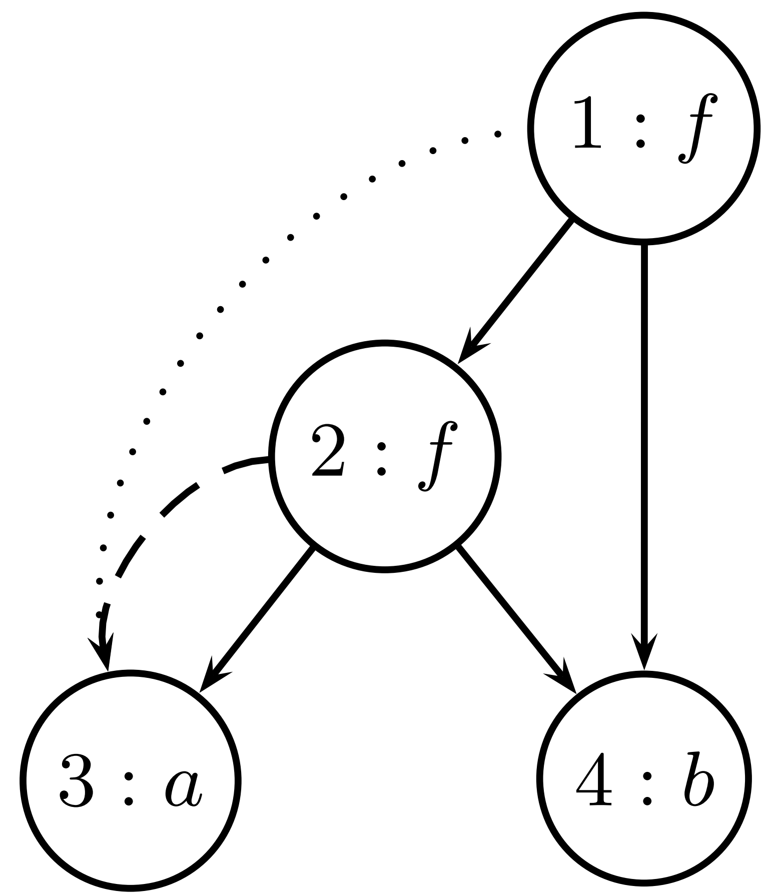
node 2 = {
  id = 2;
  fn = f;
  args = [3; 4];
  find = 3;
  ccpar = $\varnothing$;
}

node 3 = {
  id = 3;
  fn = a;
  args = [];
  find = 3;
  ccpar = $\{1, 2\}$;
}

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

Union-Find Algorithm - NODE

NODE i returns the node n with id i



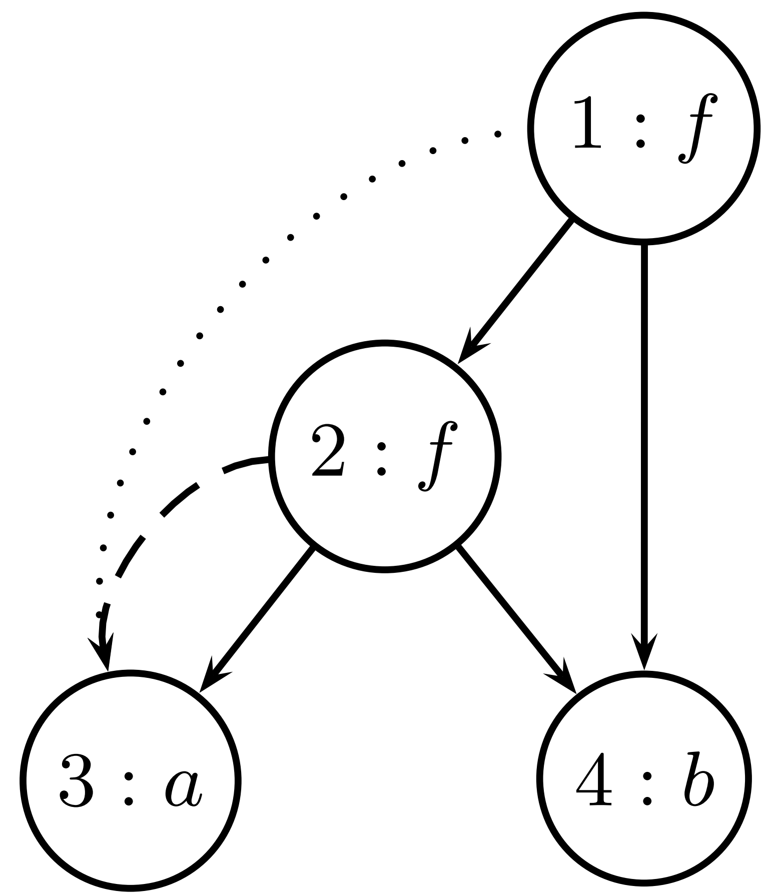
$$(\text{NODE } i).\text{id} = i$$
$$(\text{NODE } 2).\text{find} = 3$$

Union-Find Algorithm - FIND

let rec FIND $i =$

let $n =$ NODE i in

if n.find $= i$ then i else FIND n.find



FIND $2 = 3$

FIND $1 = 3$

Union-Find Algorithm - UNION

let UNION i_1 i_2 =

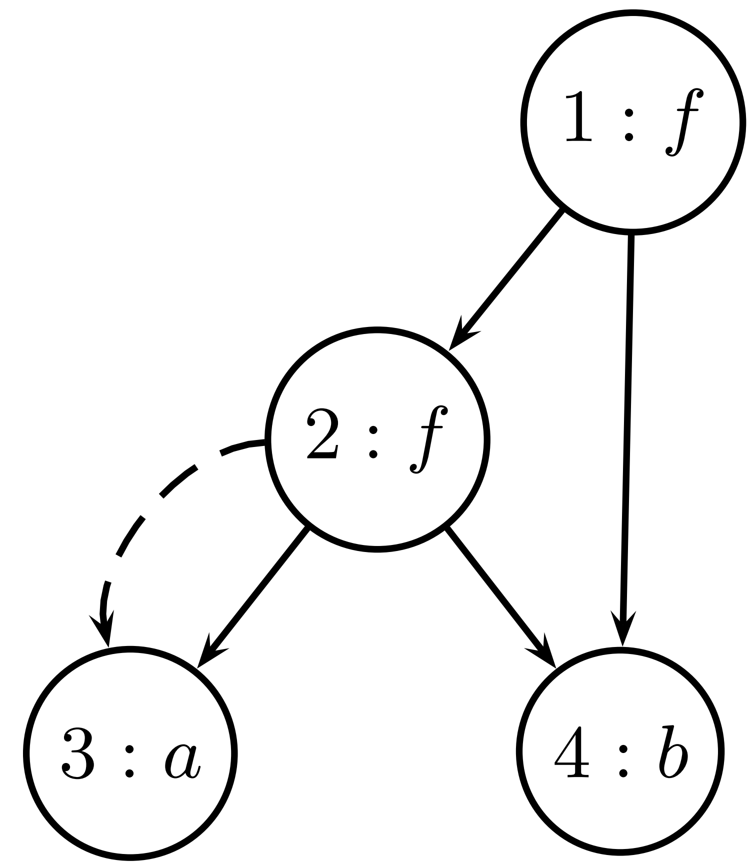
let n_1 = NODE (FIND i_1) in

let n_2 = NODE (FIND i_2) in

n_1.find $\leftarrow$ n_2.find;

n_2.ccpar $\leftarrow$ n_1.ccpar $\cup$ n_2.ccpar;

n_1.ccpar $\leftarrow \varnothing$



Union-Find Algorithm - UNION

let UNION i_1 i_2 =

let n_1 = NODE (FIND i_1) in

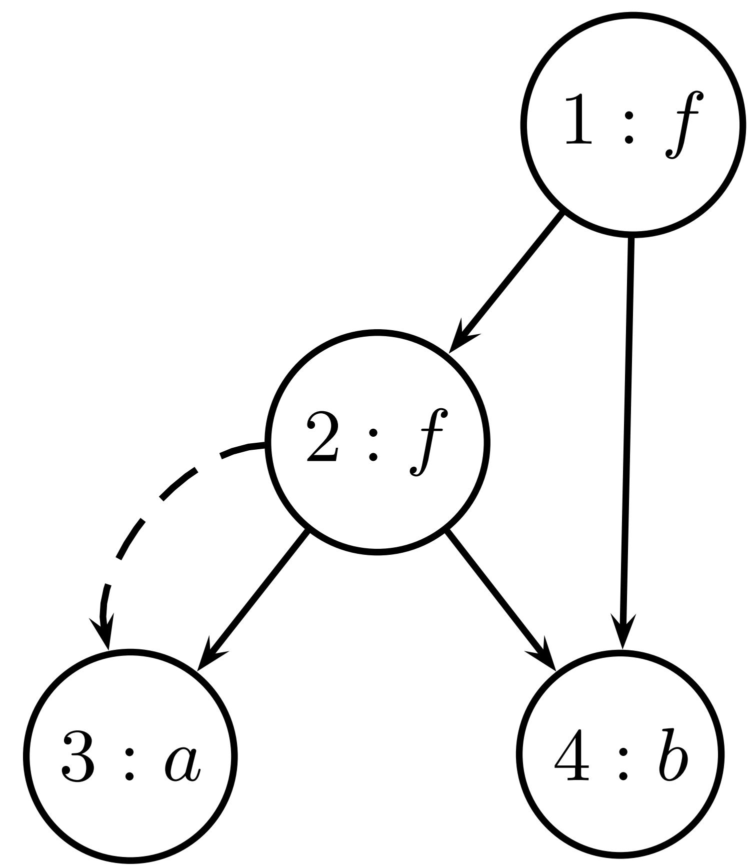
let n_2 = NODE (FIND i_2) in

n_1.find $\leftarrow$ n_2.find;

n_2.ccpar $\leftarrow$ n_1.ccpar $\cup$ n_2.ccpar;

n_1.ccpar $\leftarrow \varnothing$

UNION 1 2



Union-Find Algorithm - UNION

let UNION i_1 i_2 =

let n_1 = NODE (FIND i_1) in

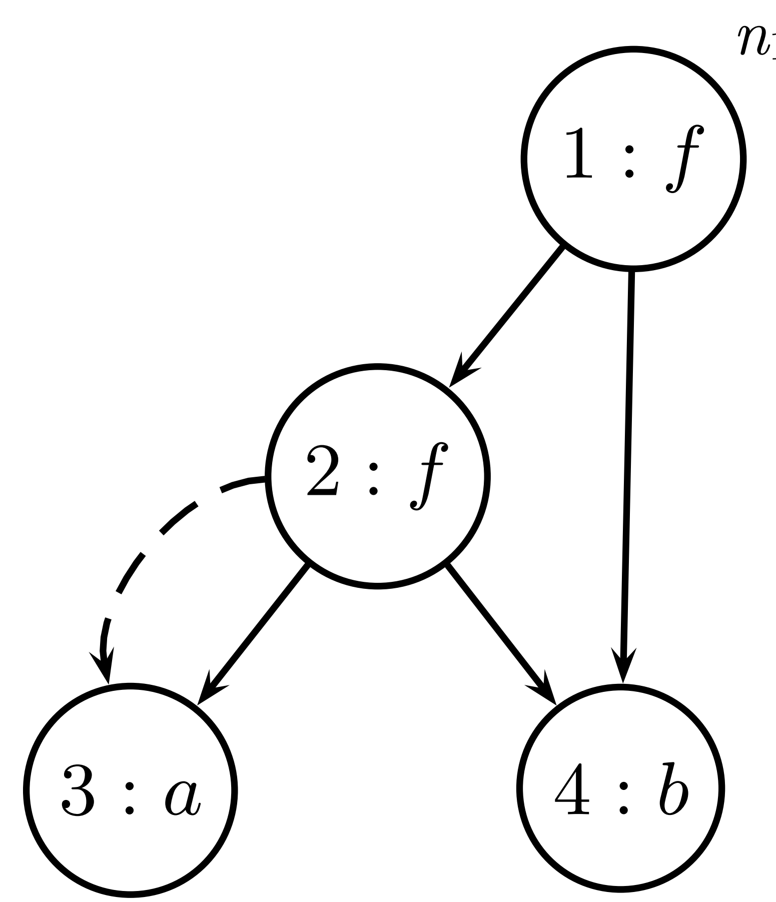
let n_2 = NODE (FIND i_2) in

n_1.find $\leftarrow$ n_2.find;

n_2.ccpar $\leftarrow$ n_1.ccpar $\cup$ n_2.ccpar;

n_1.ccpar $\leftarrow$ $\varnothing$

UNION 1 2



Union-Find Algorithm - UNION

let UNION i_1 i_2 =

let n_1 = NODE (FIND i_1) in

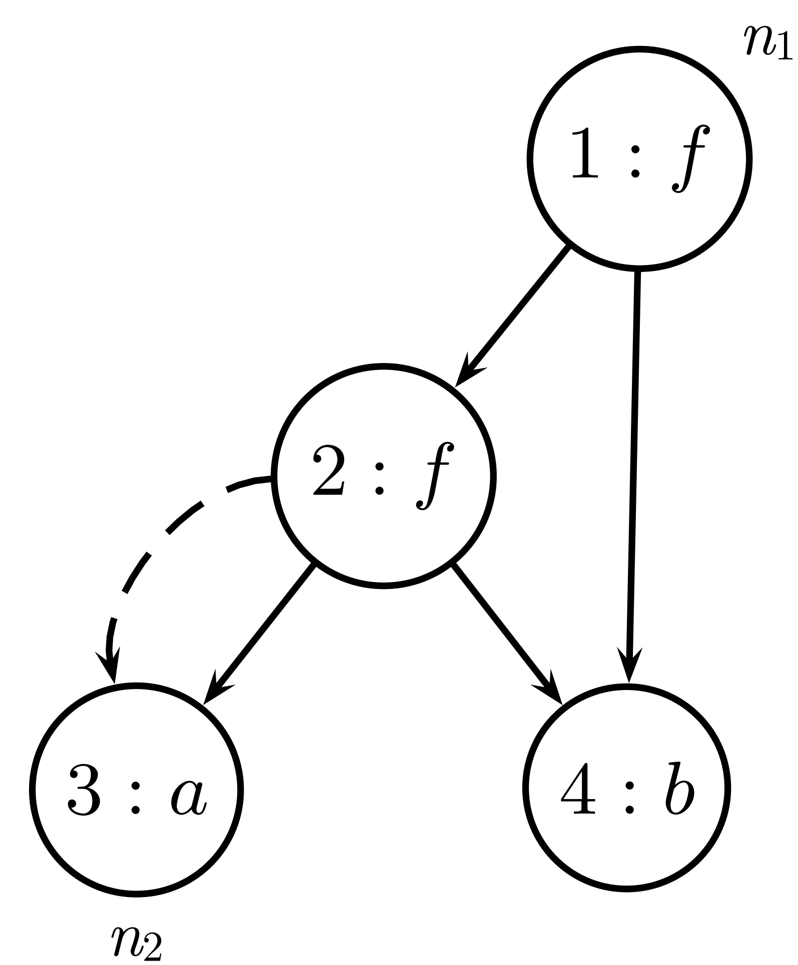
let n_2 = NODE (FIND i_2) in

n_1.find $\leftarrow$ n_2.find;

n_2.ccpar $\leftarrow$ n_1.ccpar $\cup$ n_2.ccpar;

n_1.ccpar $\leftarrow \emptyset$

UNION 1 2



Union-Find Algorithm - UNION

let UNION i_1 i_2 =

let n_1 = NODE (FIND i_1) in

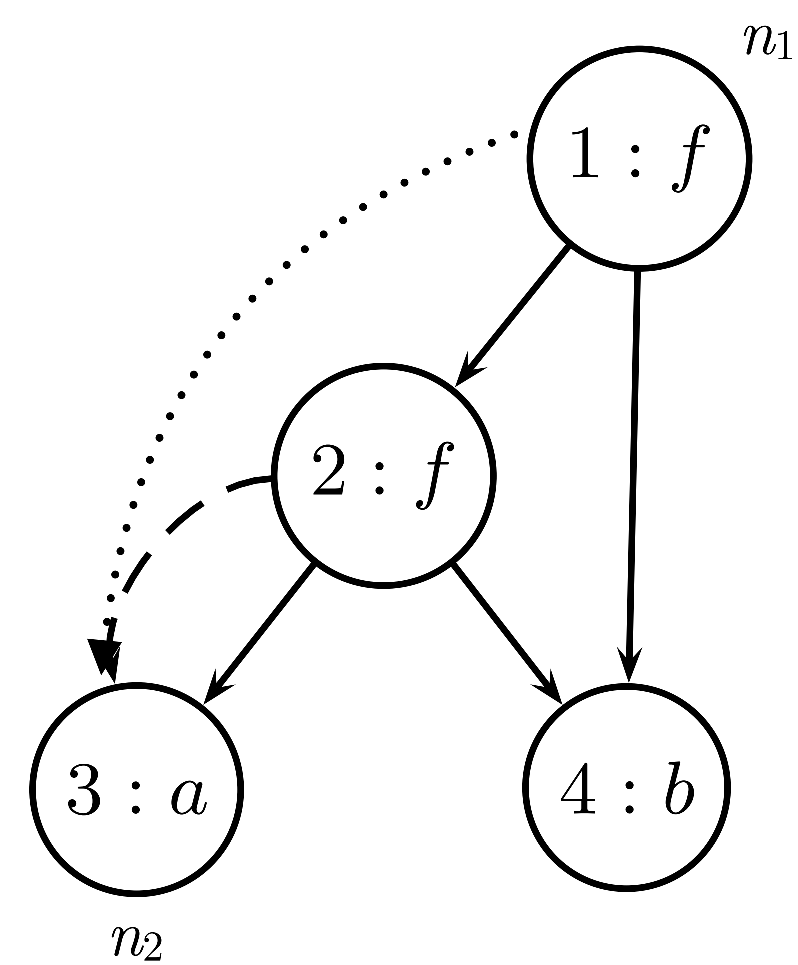
let n_2 = NODE (FIND i_2) in

n_1.find $\leftarrow$ n_2.find;

n_2.ccpar $\leftarrow$ n_1.ccpar $\cup$ n_2.ccpar;

n_1.ccpar $\leftarrow \varnothing$

UNION 1 2



Union-Find Algorithm - CCPAR

let CCPAR i =

(NODE (FIND i)).ccpar

Congruence Closure Algorithm - CONGRUENT

let CONGRUENT i_1 i_2 =

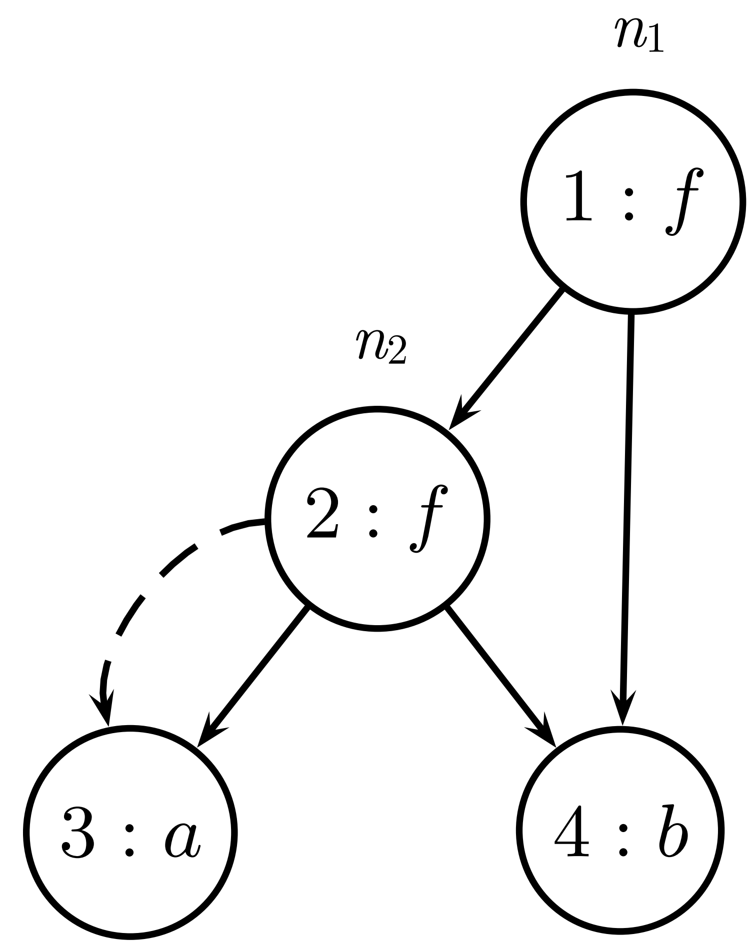
let n_1 = NODE i_1 in

let n_2 = NODE i_2 in

$n_1.\mathrm{fn} = n_2.\mathrm{fn}$

$\wedge\ |n_1.\mathrm{args}| = |n_2.\mathrm{args}|$

$\wedge\ \forall i \in \{1, ..., |n_1.\mathrm{args}|\}.\ \mathrm{FIND}\ n_1.\mathrm{args}[i] = \mathrm{FIND}\ n_2.\mathrm{args}[i]$



Congruence Closure Algorithm - MERGE

let rec MERGE i_1 i_2 =

if FIND $i_1 \neq$ FIND i_2 then begin

let P_1 = CCPAR i_1 in

let P_2 = CCPAR i_2 in

UNION i_1 i_2;

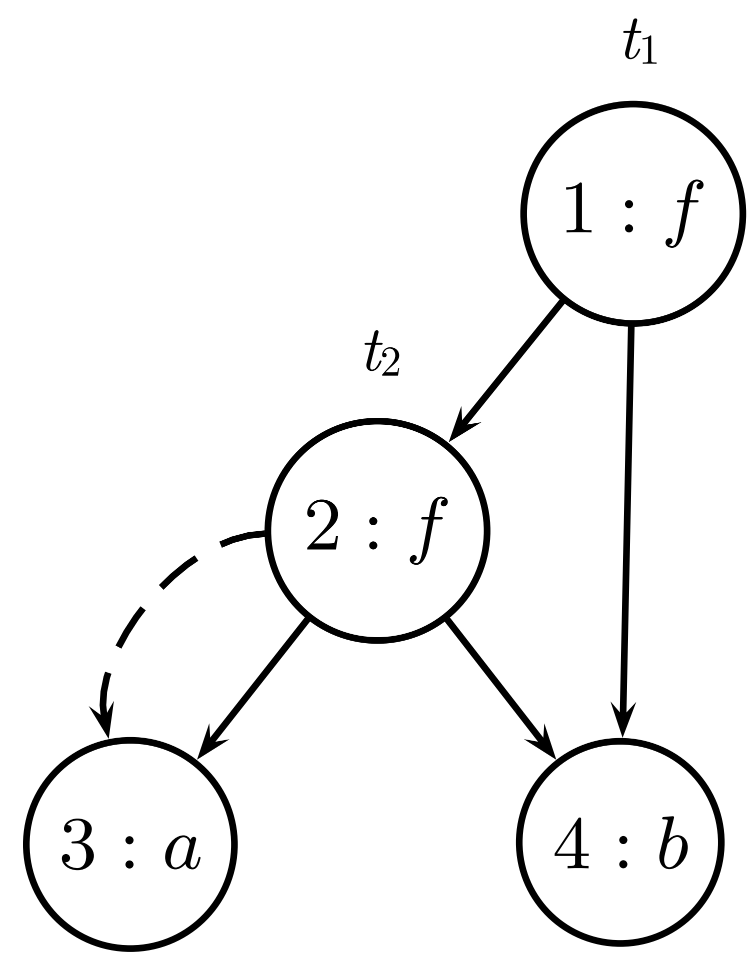
foreach $t_1, t_2 \in P_1 \times P_2$ do

if FIND $t_1 \neq$ FIND $t_2 \wedge$ CONGRUENT t_1 t_2

then MERGE t_1 t_2

done

end



Decision Procedure for T_E-Satisfiability

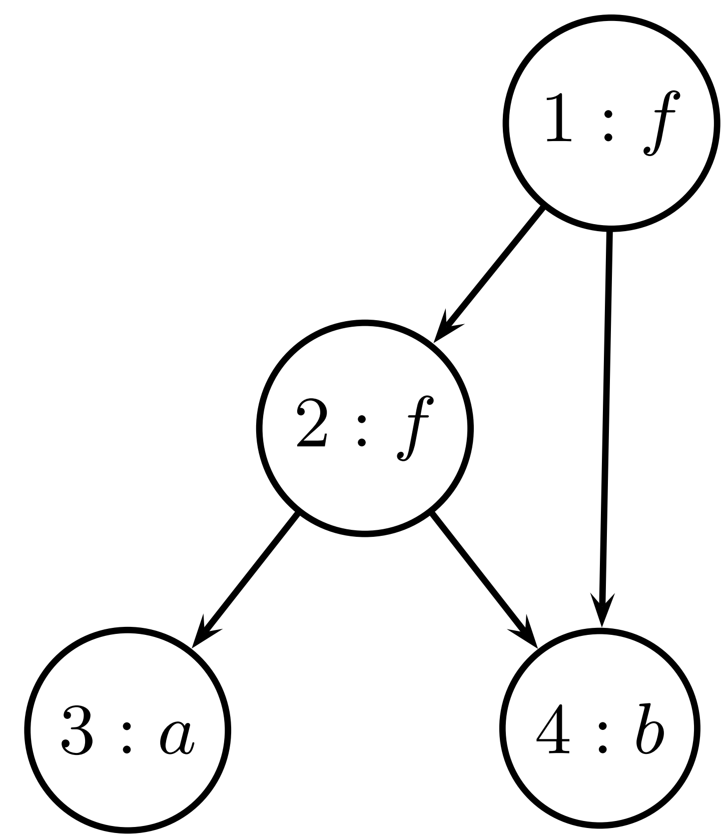
$$F : s_1 = t_1 \wedge \ldots \wedge s_m = t_m \wedge s_{m+1} \neq t_{m+1} \wedge \ldots \wedge s_n \neq t_n$$

1. Construct the initial DAG for the subterm set S_F
2. For $i \in \{1, \ldots, m\}$, $\mathrm{MERGE}\ s_i\ t_i$
3. If $\mathrm{FIND}\ s_i = \mathrm{FIND}\ t_i$ for some i $\in \{m + 1, \ldots, n\}$, return unsatisfiable
4. Otherwise, return satisfiable

Deciding T_E-Satisfiability Example 1

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

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

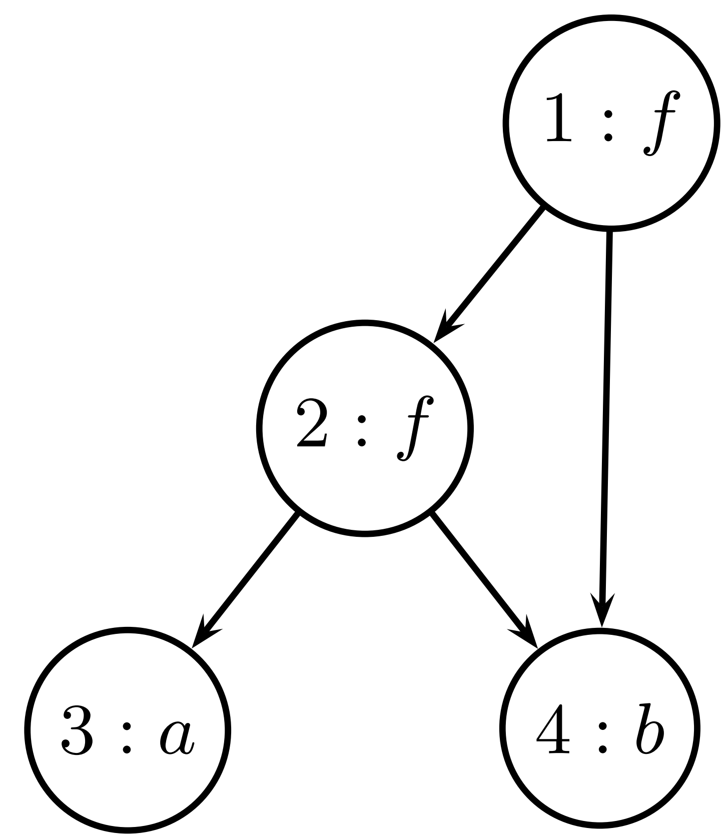


Deciding T_E-Satisfiability Example 1

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

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

1. MERGE 2 3



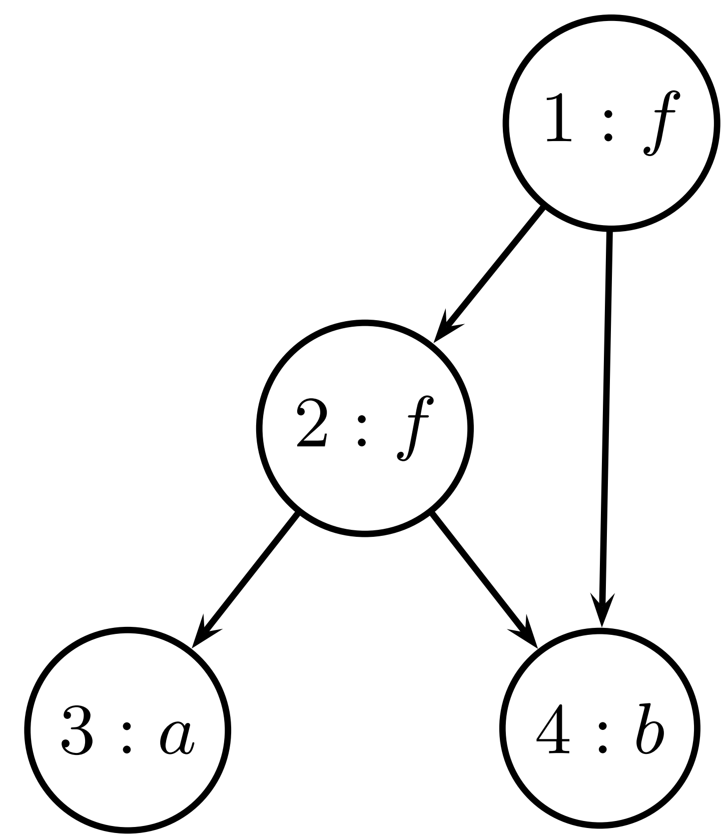
Deciding T_E-Satisfiability Example 1

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

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

1. MERGE 2 3

   (1) $P_2 = \text{CCPAR } 2 = \{1\}$

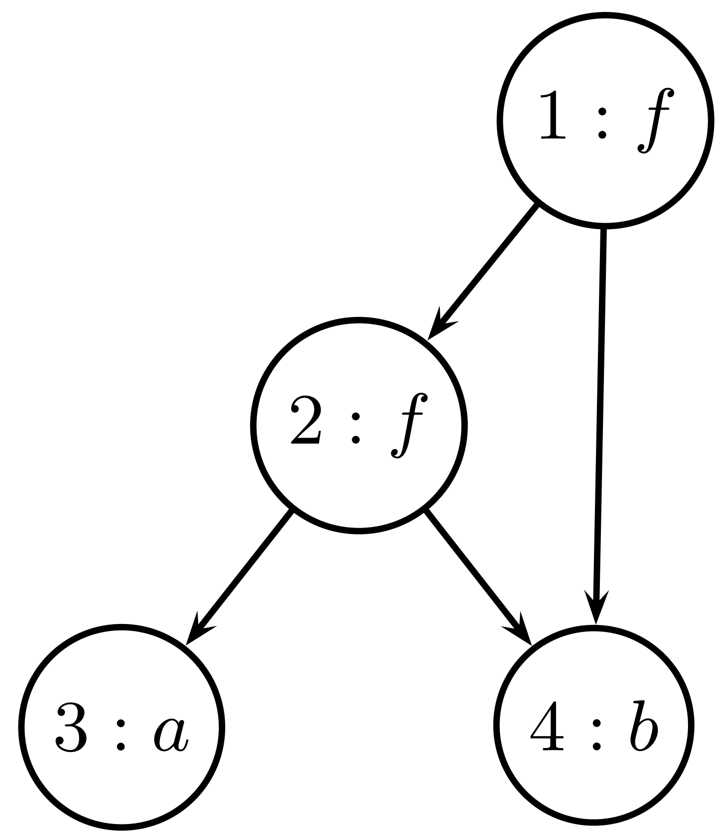


Deciding T_E-Satisfiability Example 1

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

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

1. MERGE 2 3
   (1) $P_2 = \text{CCPAR } 2 = \{1\}$
   (2) $P_3 = \text{CCPAR } 3 = \{2\}$

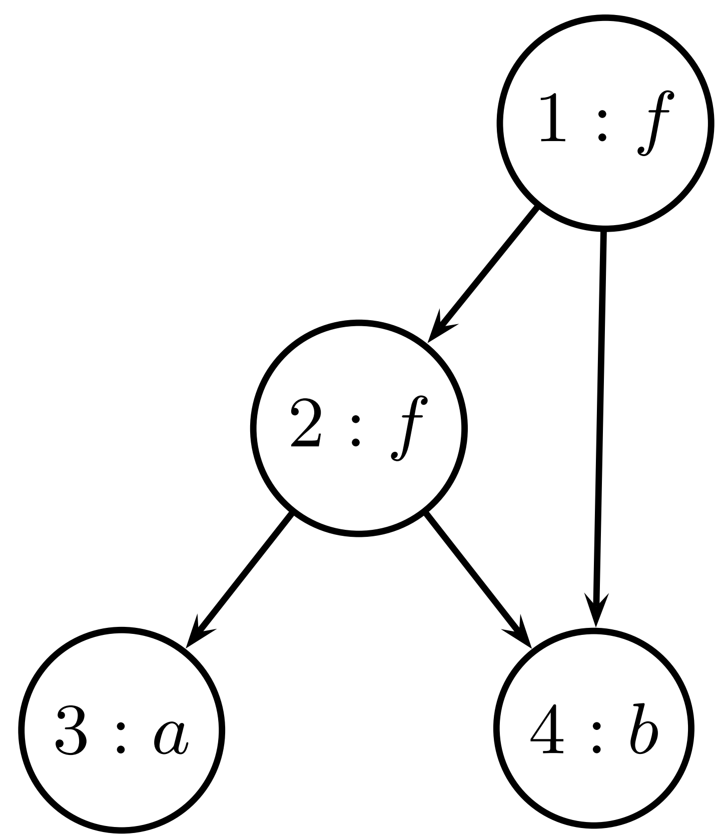


Deciding T_E-Satisfiability Example 1

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

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

1. MERGE 2 3
   (1) $P_2 =$ CCPAR $2 = \{1\}$
   (2) $P_3 =$ CCPAR $3 = \{2\}$
   (3) UNION 2 3

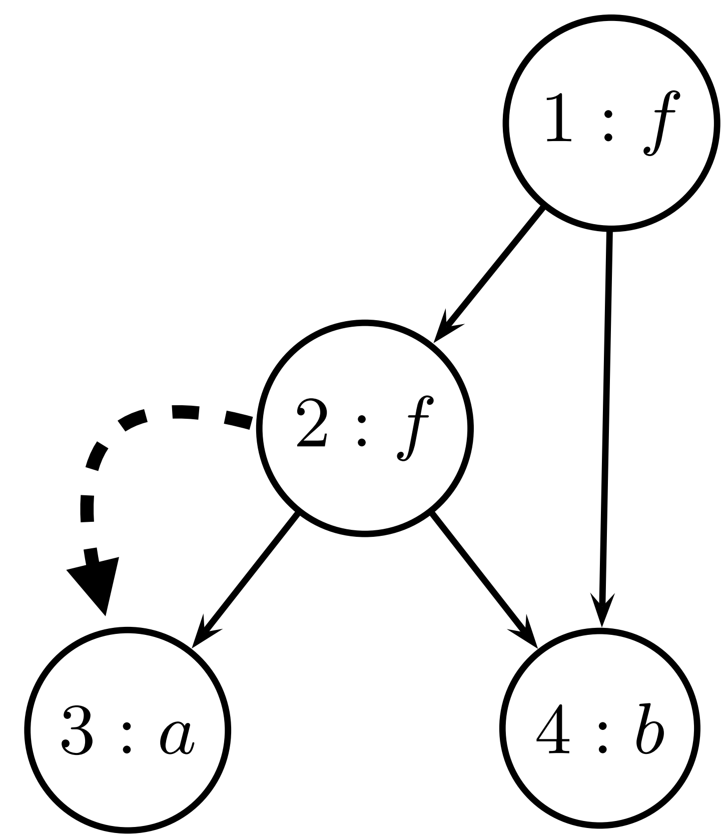


Deciding T_E-Satisfiability Example 1

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

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

1. MERGE 2 3
   (1) $P_2 = \text{CCPAR } 2 = \{1\}$
   (2) $P_3 = \text{CCPAR } 3 = \{2\}$
   (3) UNION 2 3

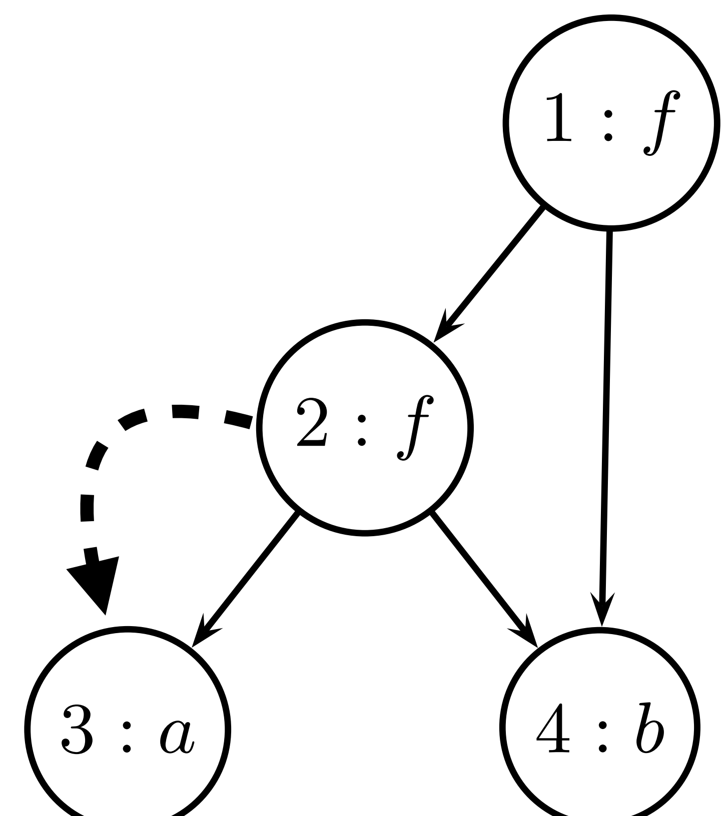


Deciding T_E-Satisfiability Example 1

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

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

1. MERGE 2 3
   (1) $P_2 = \text{CCPAR } 2 = \{1\}$
   (2) $P_3 = \text{CCPAR } 3 = \{2\}$
   (3) UNION 2 3
   (4) MERGE 1 2

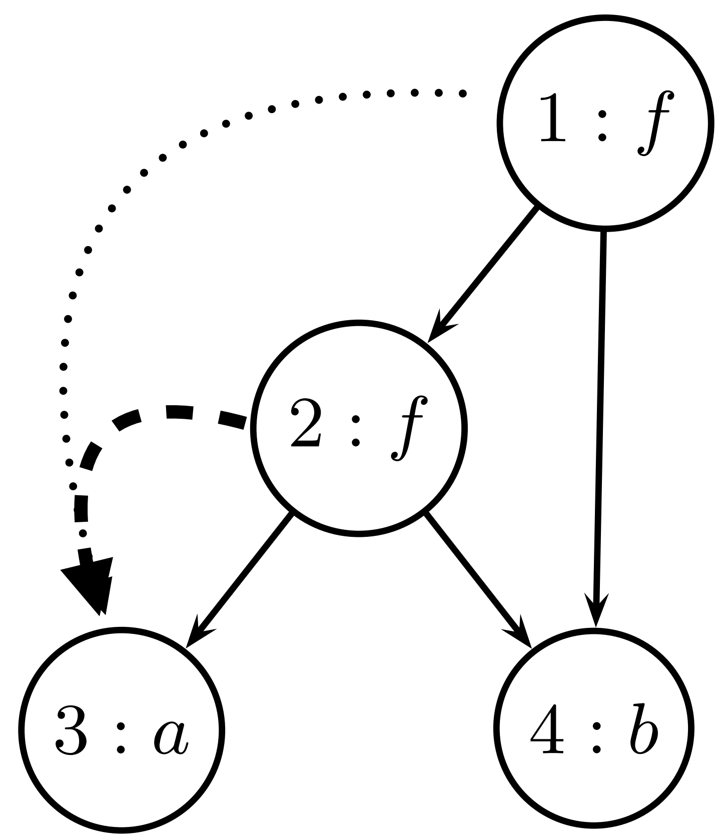


Deciding T_E-Satisfiability Example 1

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

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

1. MERGE 2 3
   (1) P_2 = CCPAR 2 = $\{1\}$
   (2) P_3 = CCPAR 3 = $\{2\}$
   (3) UNION 2 3
   (4) MERGE 1 2



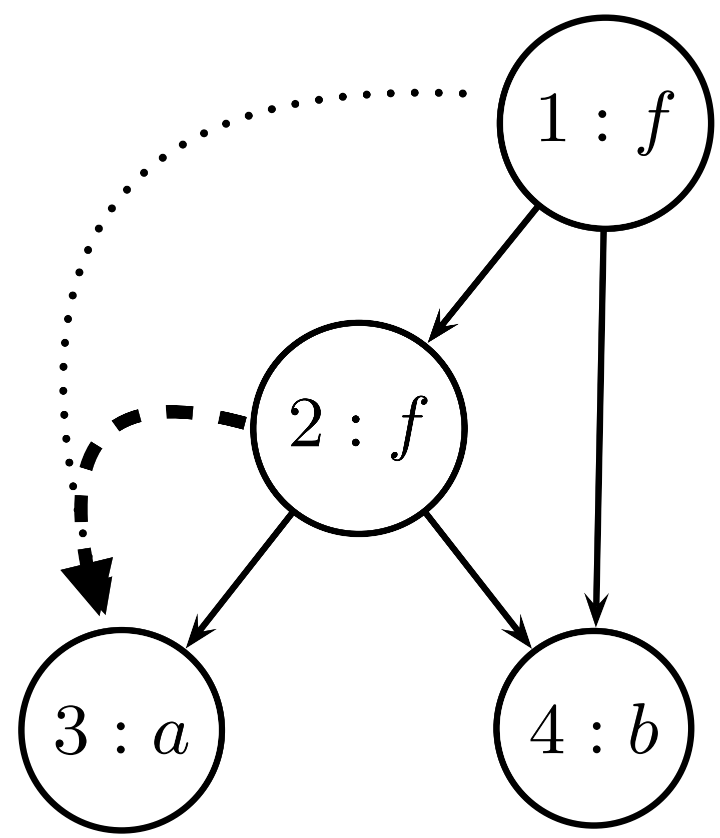
Deciding T_E-Satisfiability Example 1

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

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

1. MERGE 2 3
   (1) P_2 = CCPAR 2 = $\{1\}$
   (2) P_3 = CCPAR 3 = $\{2\}$
   (3) UNION 2 3
   (4) MERGE 1 2

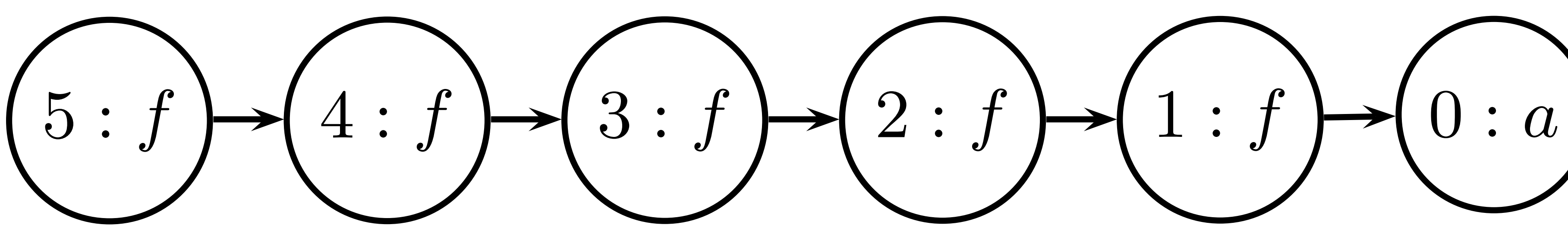
T_E-unsatisfiable



Deciding T_E-Satisfiability Example 2

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

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



Deciding T_E-Satisfiability Example 2

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

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

1. MERGE 3 0

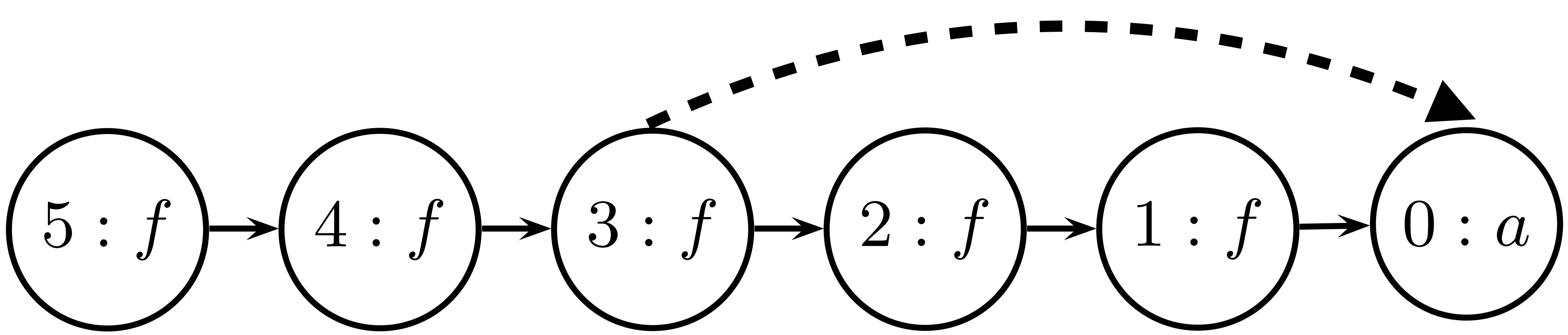
$$\boxed{5 : f} \rightarrow \boxed{4 : f} \rightarrow \boxed{3 : f} \rightarrow \boxed{2 : f} \rightarrow \boxed{1 : f} \rightarrow \boxed{0 : a}$$

Deciding T_E-Satisfiability Example 2

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

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

1. MERGE 3 0

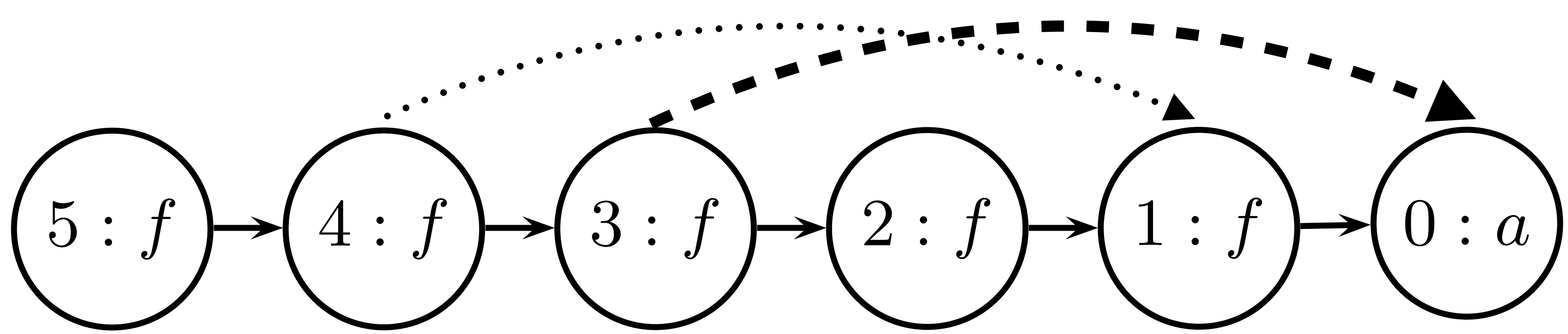


Deciding T_E-Satisfiability Example 2

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

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

1. MERGE 3 0

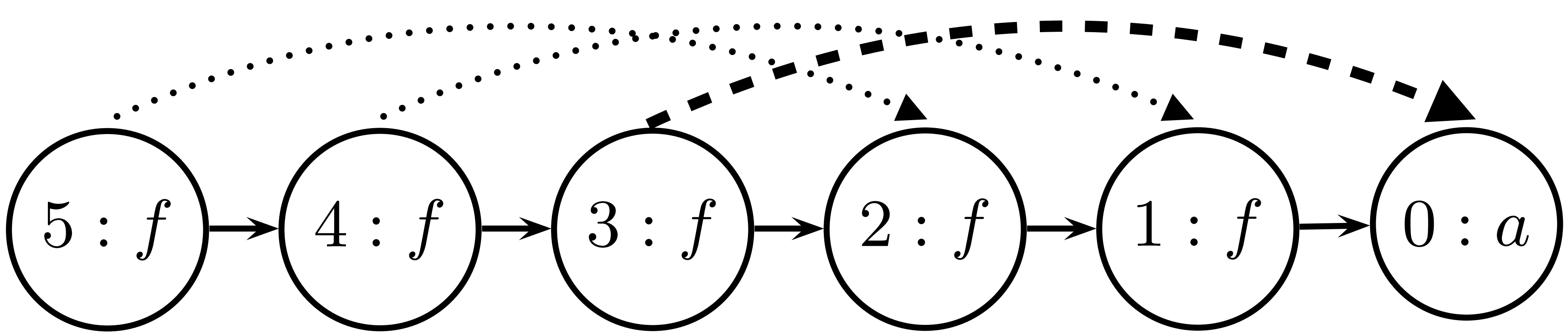


Deciding $\mathsf{T_E}$-Satisfiability Example 2

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

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

1. MERGE 3 0

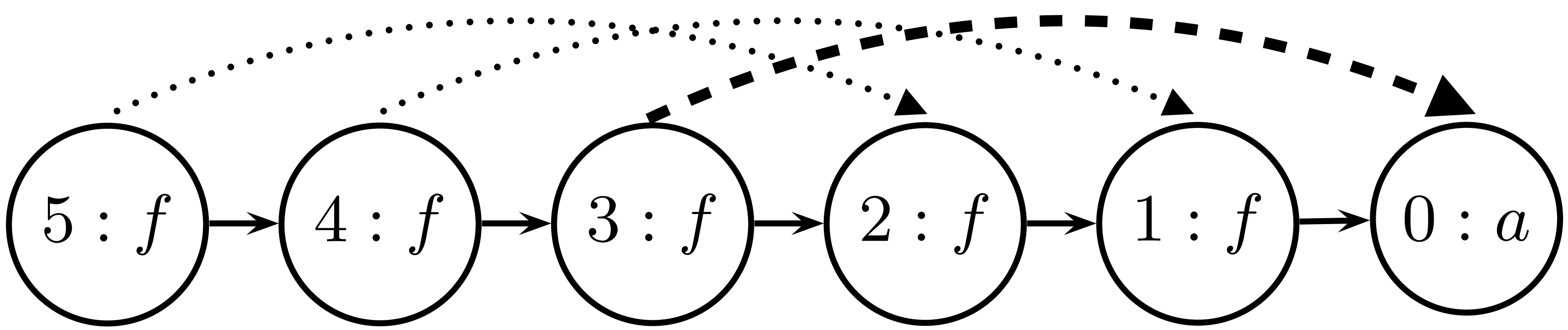


Deciding T_E-Satisfiability Example 2

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

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

1. MERGE 3 0
2. MERGE 5 0

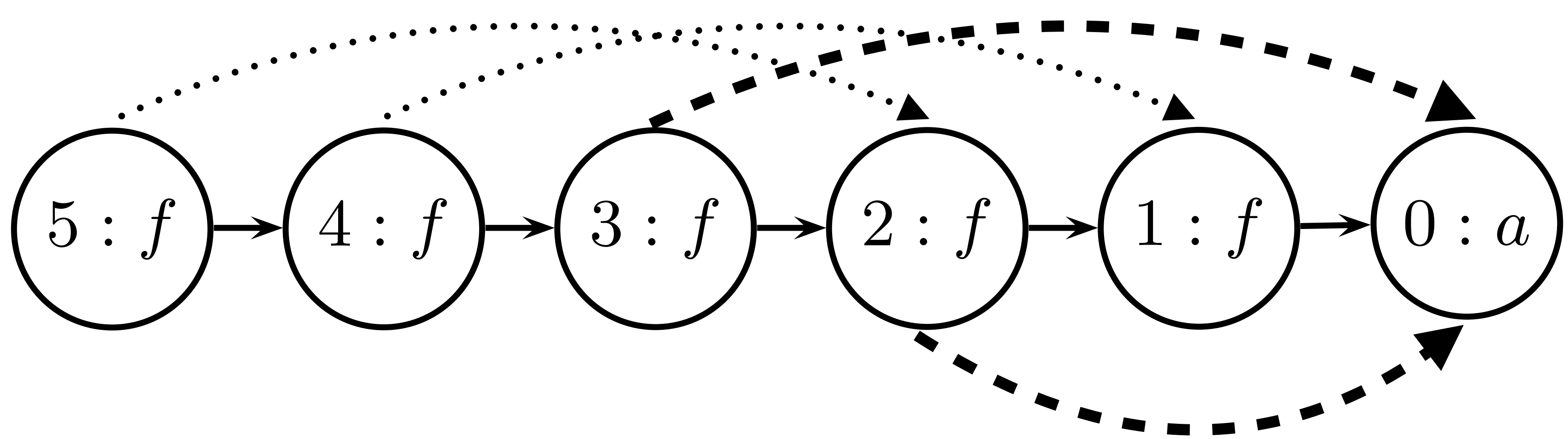


Deciding T_E-Satisfiability Example 2

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

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

1. MERGE 3 0
2. MERGE 5 0

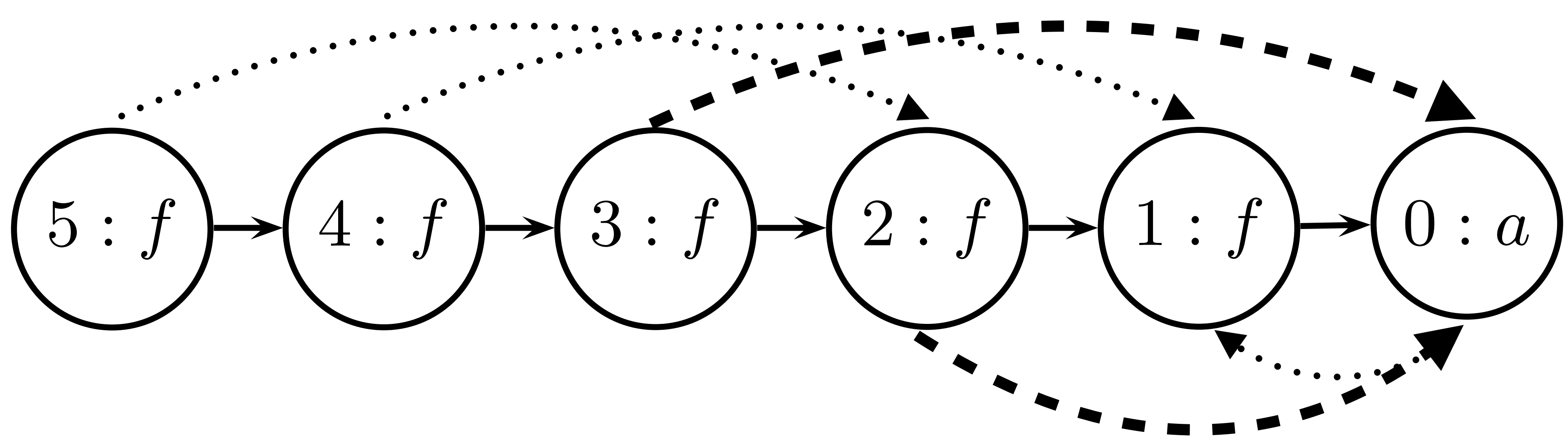


Deciding T_E-Satisfiability Example 2

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

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

1. MERGE 3 0
2. MERGE 5 0

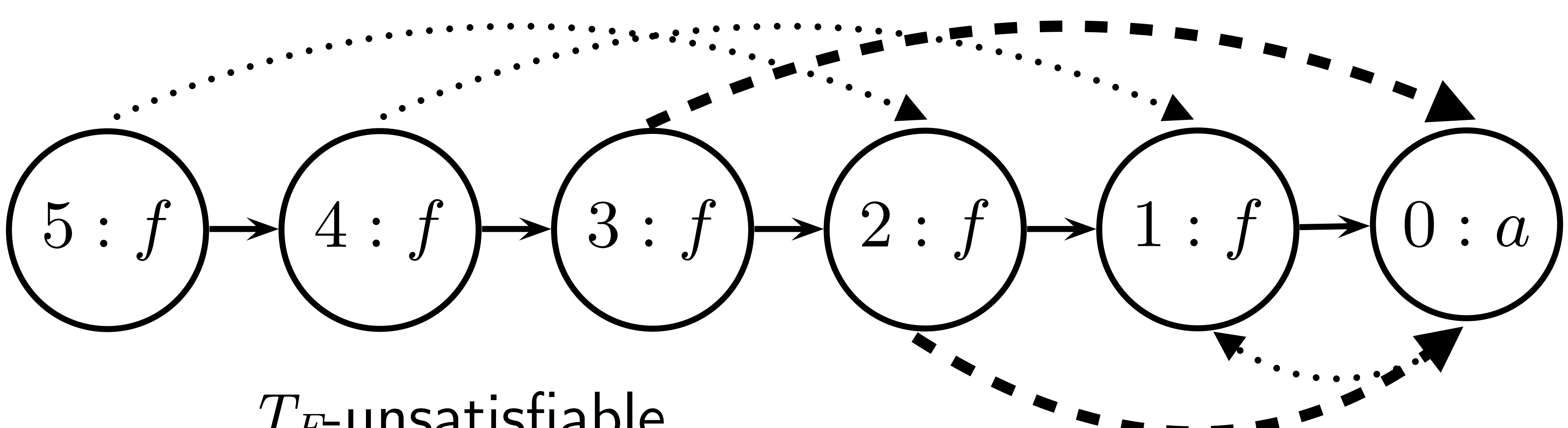


Deciding T_E-Satisfiability Example 2

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

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

1. MERGE 3 0
2. MERGE 5 0



T_E-unsatisfiable

Soundness and Completeness

Theorem (Sound & Complete). Quantifier-free conjunctive Σ_E-formula F is T_E-satisfiable iff the congruence closure algorithm returns satisfiable

Complexity

Let e be the number of edges and n be the number of nodes in the initial DAG.

Theorem (Complexity). The congruence closure algorithm run in time $O(e^2)$ for $O(n)$ MERGEs.

Recursive Data Structures

T_{RDS}

- Can model
  - records
  - lists
  - trees
  - stacks
- Cannot model
  - queues

Theory of Lists - T_{cons}

$$\Sigma_{cons} : \{cons,\ car,\ cdr,\ atom,\ =\}$$

- $cons$: a binary function, called the constructor;
- car: a unary function, called the left projector;
- cdr: a unary function, called the right projector;
- $atom$: a unary predicate;
- $=$: a binary predicate

$$car(cons(a,\ b)) = a$$
$$cdr(cons(a,\ b)) = b$$

Axioms of T_{cons}

- Axioms of (reflexivity), (symmetry), and (transitivity) of T_E
- Instantiations of the (function congruence) axiom schema for $cons$, car, and cdr:
  - $\forall x_1, x_2, y_1, y_2.\ x_1 = x_2 \wedge y_1 = y_2 \rightarrow cons(x_1, y_1) = cons(x_2, y_2)$
  - $\forall x, y.\ x = y \rightarrow car(x) = car(y)$
  - $\forall x, y.\ x = y \rightarrow cdr(x) = cdr(y)$
- An instantiation of the (predicate congruence) axiom schema for atom:
  - $\forall x, y.\ x = y \rightarrow (atom(x) \leftrightarrow atom(y))$

Axioms of T_{cons}

- $\forall x, y.\ car(cons(x,\ y)) = x$ (left projection)
- $\forall x, y.\ cdr(cons(x,\ y)) = y$ (right projection)
- $\forall x.\ \neg atom(x) \rightarrow cons(car(x),\ cdr(x)) = x$ (construction)
- $\forall x, y.\ \neg atom(cons(x,\ y))$ (atom)

Decidability

- T_{cons}: undecidable
- quantifier-free T_{cons}: decidable

Preprocess

By the (construction) axiom, replace

$$\neg atom(u_i)$$

with

$$u_i = cons(u_i^1,\ u_i^2)$$

$$\forall x.\ \neg atom(x) \rightarrow cons(car(x),\ cdr(x)) = x \qquad \text{(construction)}$$

Decision Procedure

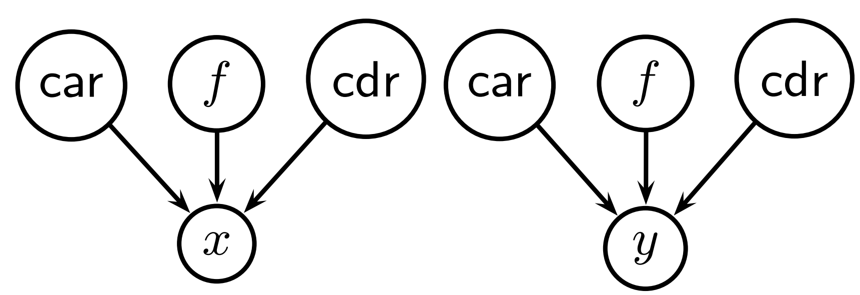
$$F : s_1 = t_1 \wedge \ldots \wedge s_m = t_m \wedge s_{m+1} \neq t_{m+1} \wedge \ldots \wedge s_n \neq t_n$$
$$\wedge\ atom(u_1) \wedge \ldots \wedge atom(u_l)$$

- Construct the initial DAG for the subterm set S_F
- By the (left projection) and (right projection) axioms, for each node n such that $n.\mathrm{fn} = cons$,
  - add $car(n)$ to the DAG and $\mathrm{MERGE}\ car(n)\ n.\mathrm{args}[1]$;
  - add $cdr(n)$ to the DAG and $\mathrm{MERGE}\ cdr(n)\ n.\mathrm{args}[2]$;
- For $i \in \{1, \ldots, m\}$, $\mathrm{MERGE}\ s_i\ t_i$
- For $i \in \{m+1, \ldots, n\}$, if $\mathrm{FIND}\ s_i = \mathrm{FIND}\ t_i$, return unsatisfiable
- By the (atom axiom), for $i \in \{1, \ldots, l\}$, if $\exists v.\ \mathrm{FIND}\ v = \mathrm{FIND}\ u_i \wedge v.\mathrm{fn} = cons$, return unsatisfiability
- Otherwise, return satisfiable

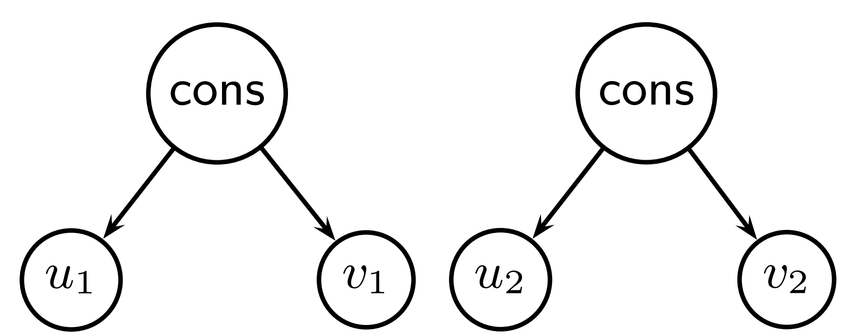
Combining T_E and $\mathrm{T}_{\mathrm{cons}}$ - Example

$$F : car(x) = car(y) \wedge cdr(x) = cdr(y) \wedge f(x) \neq f(y) \wedge \neg atom(x) \wedge \neg atom(y)$$

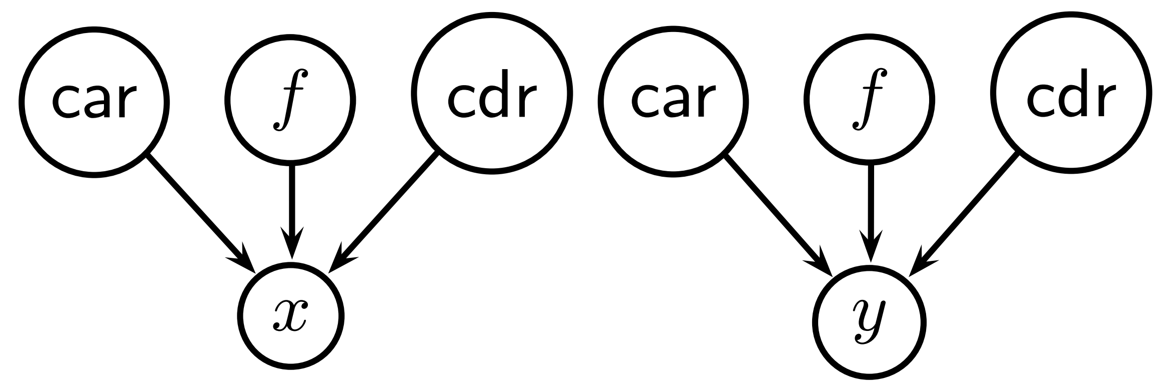
$$F' : car(x) = car(y) \wedge cdr(x) = cdr(y) \wedge f(x) \neq f(y) \wedge x = cons(u_1, v_1) \wedge y = cons(u_2, v_2)$$



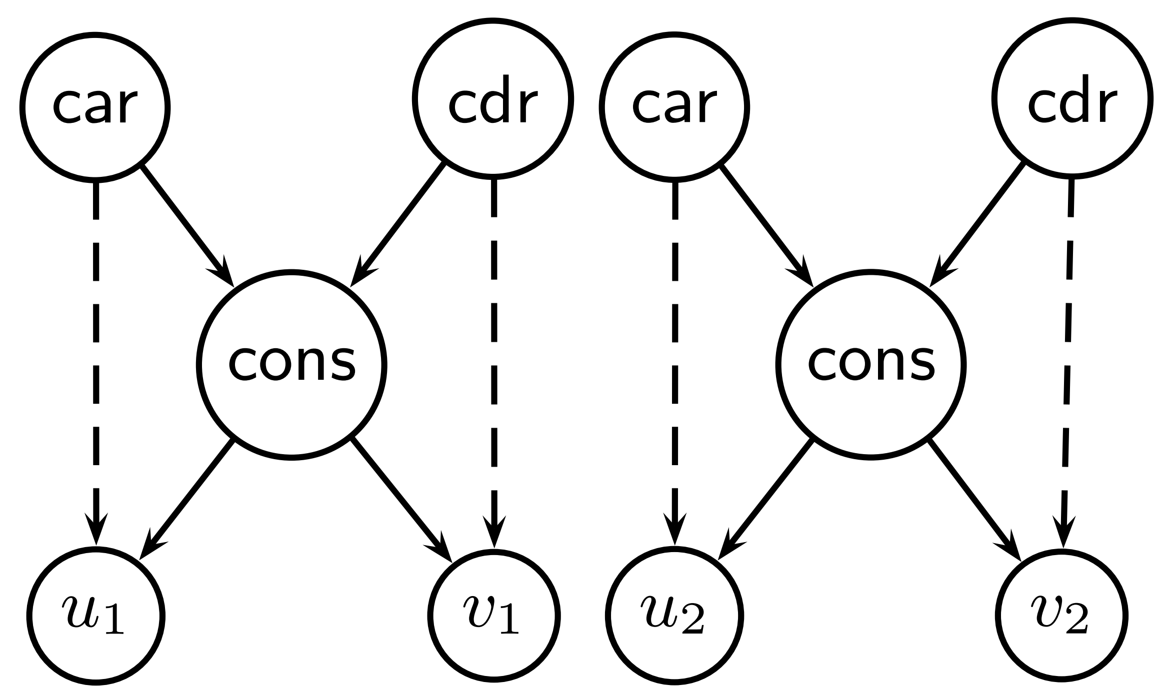
Step 1: initial DAG



Combining T_E and T_{cons} - Example

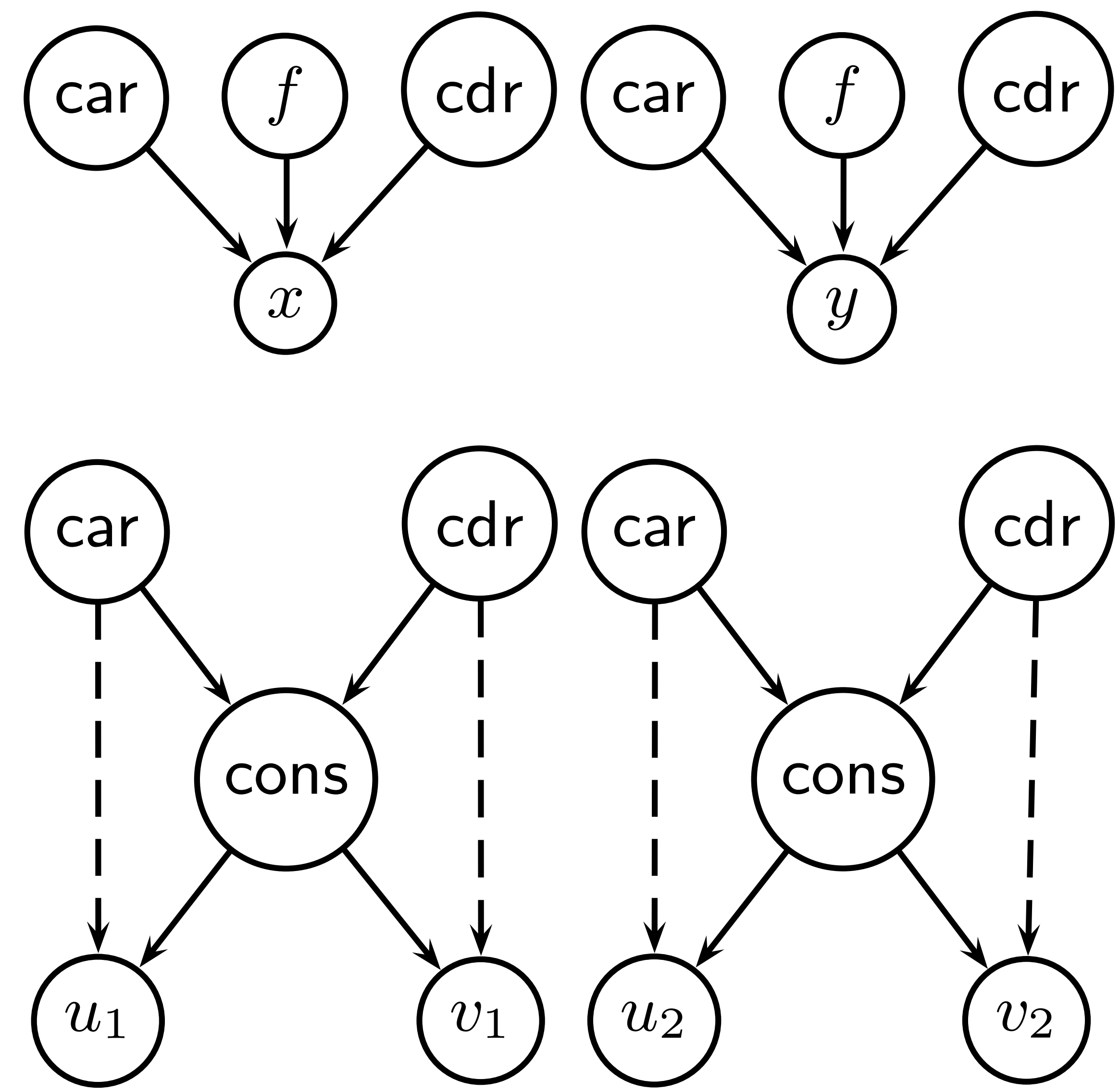


Step 2: add $car(n)$ and $cdr(n)$



Combining T_E and T_{cons} - Example

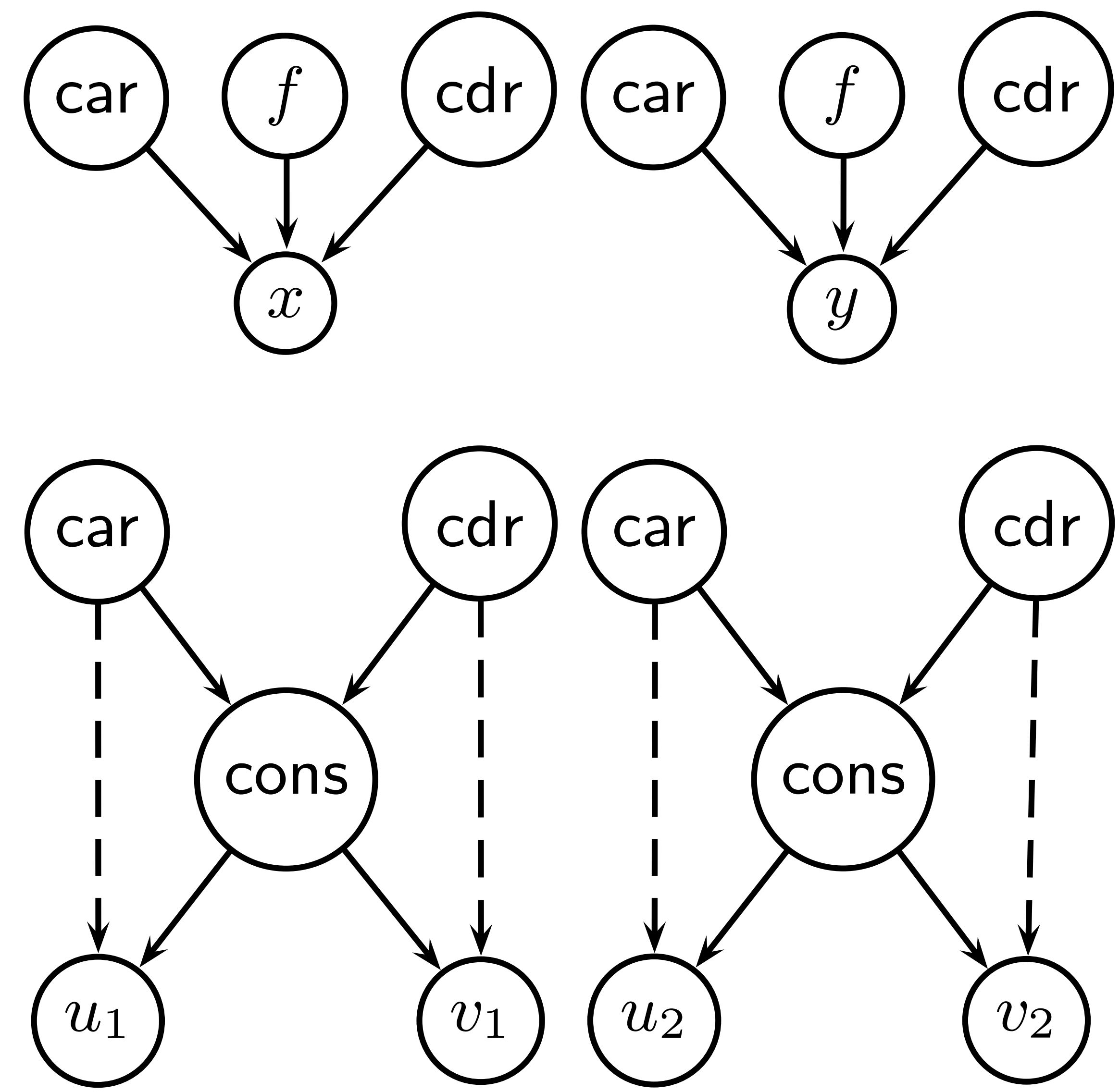
Step 3: $\text{MERGE } s_i \ t_i$



Combining T_E and T_{cons} - Example

Step 3: $\text{MERGE}\ s_i\ t_i$

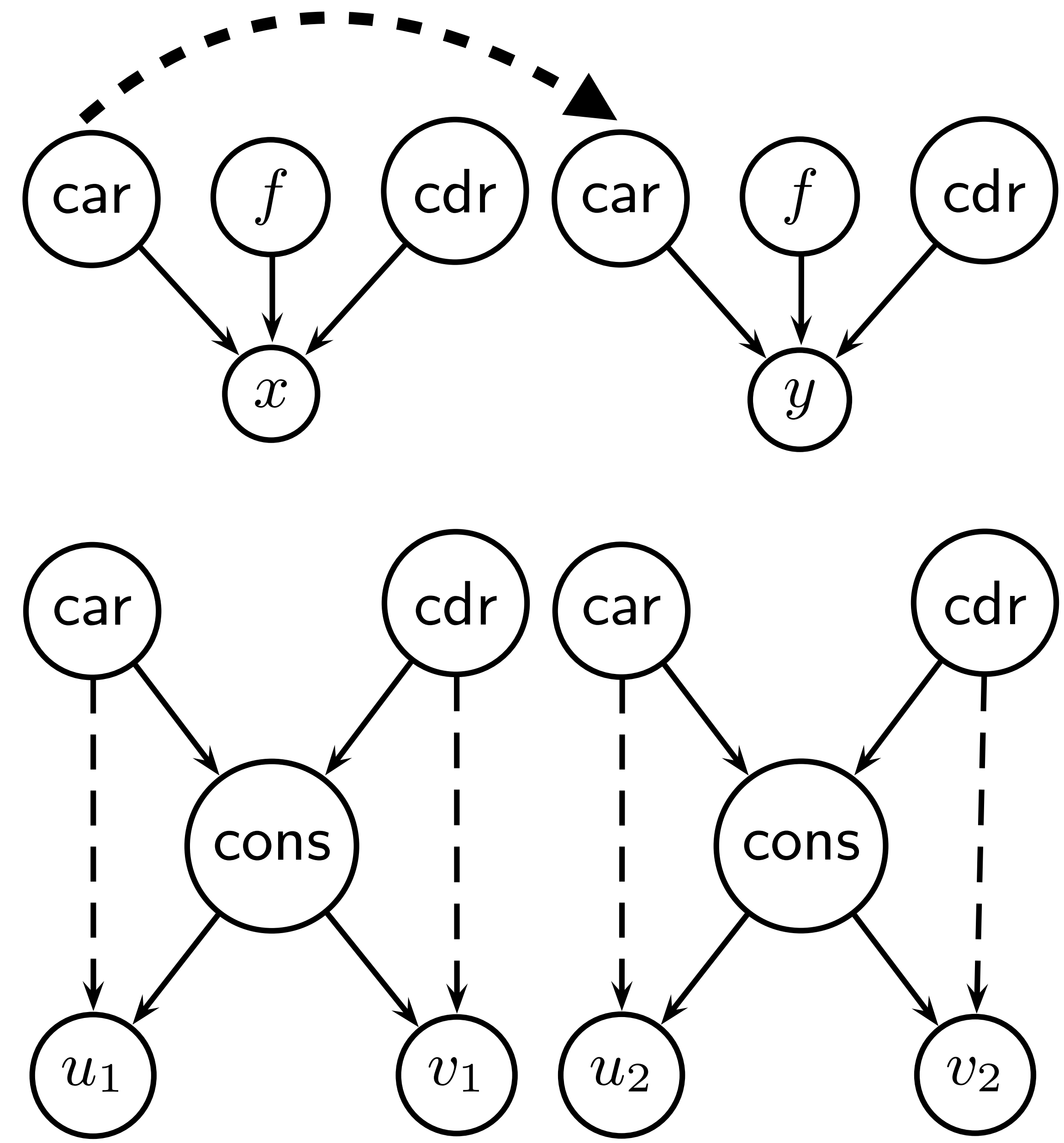
1. $car(x) = car(y)$



Combining T_E and T_{cons} - Example

Step 3: $\mathrm{MERGE}\ s_i\ t_i$

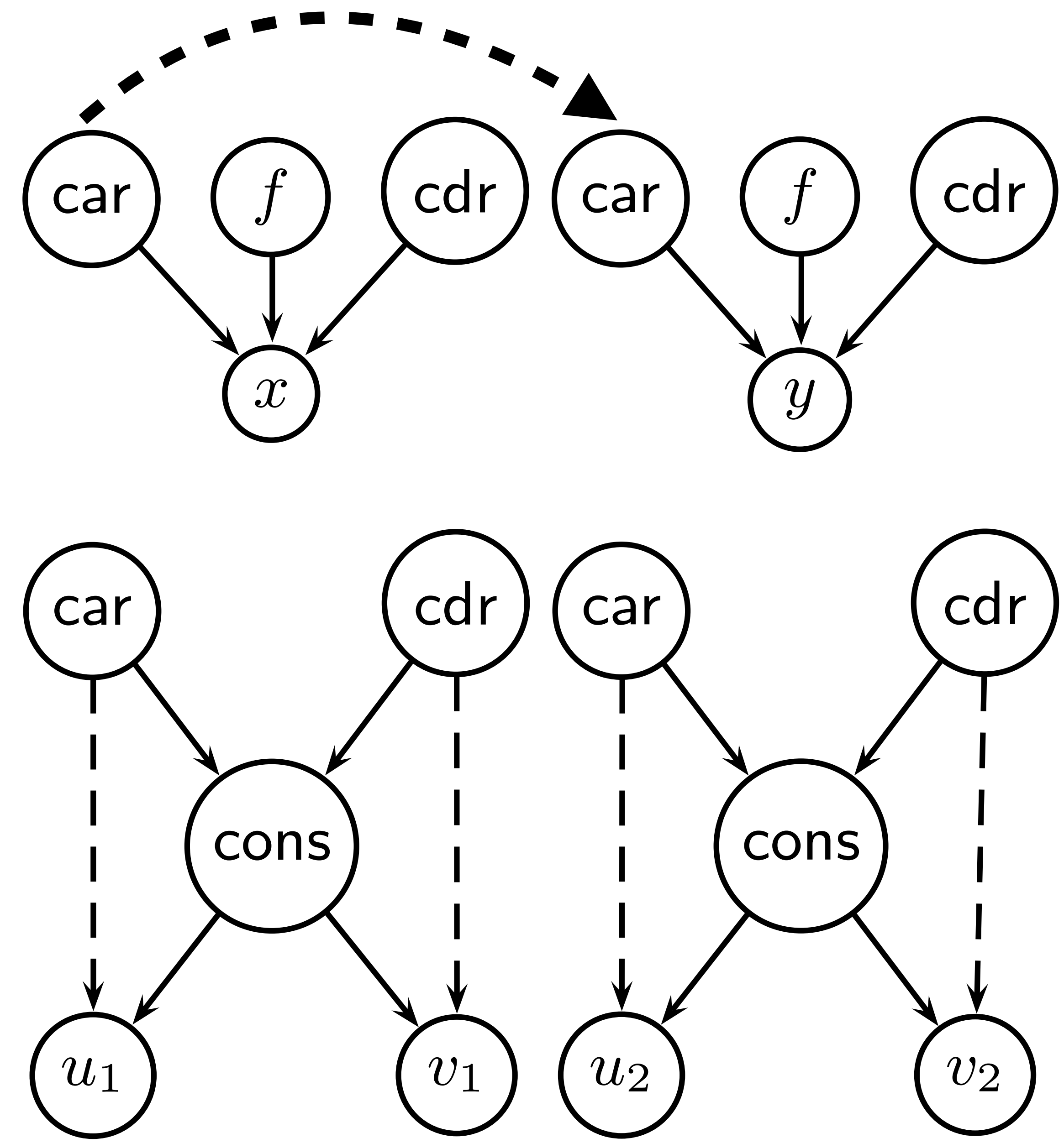
1. $car(x) = car(y)$



Combining T_E and T_{cons} - Example

Step 3: $\mathrm{MERGE}\ s_i\ t_i$

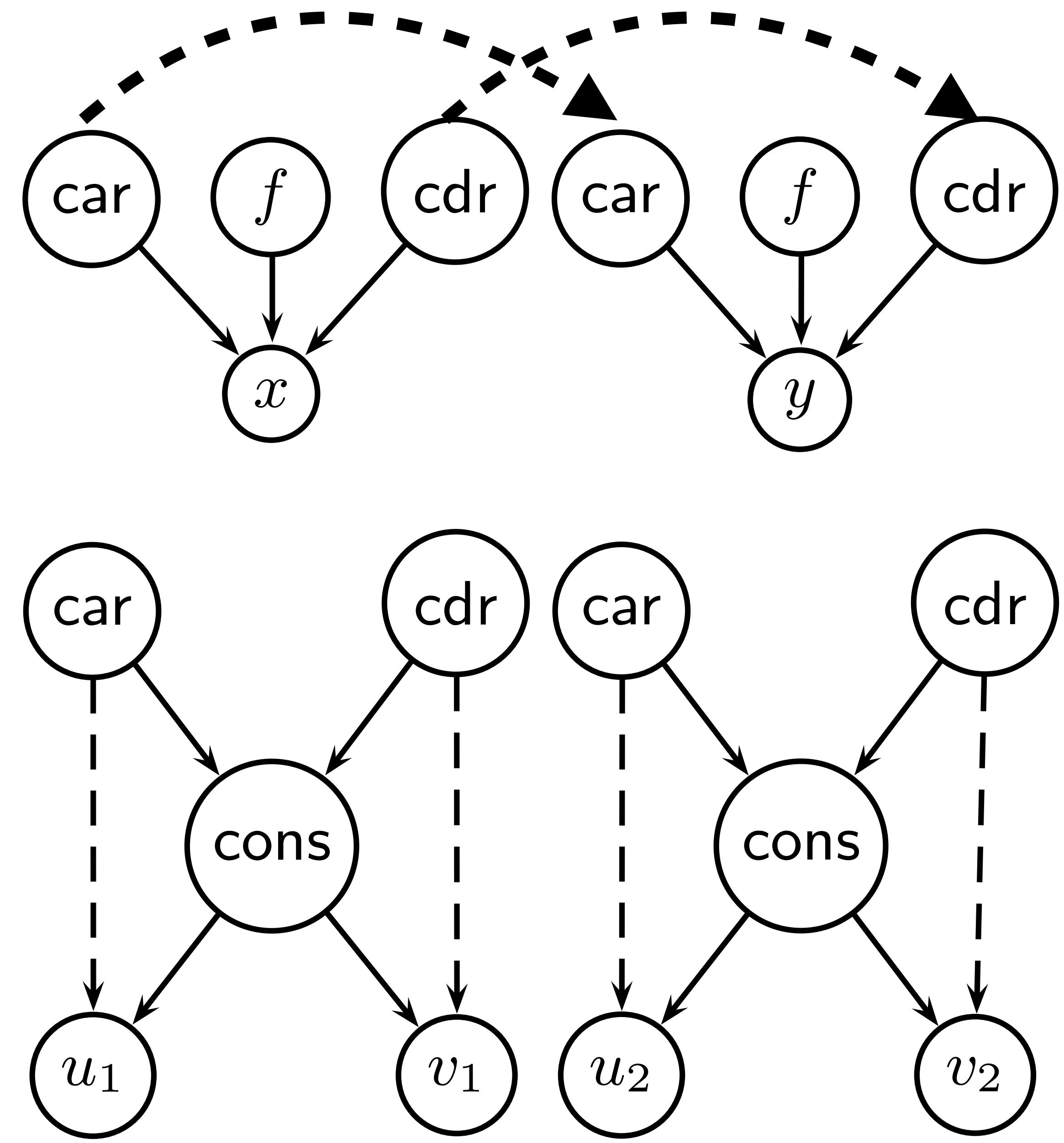
1. $car(x) = car(y)$
2. $cdr(x) = cdr(y)$



Combining T_E and T_{cons} - Example

Step 3: $\mathrm{MERGE}\ s_i\ t_i$

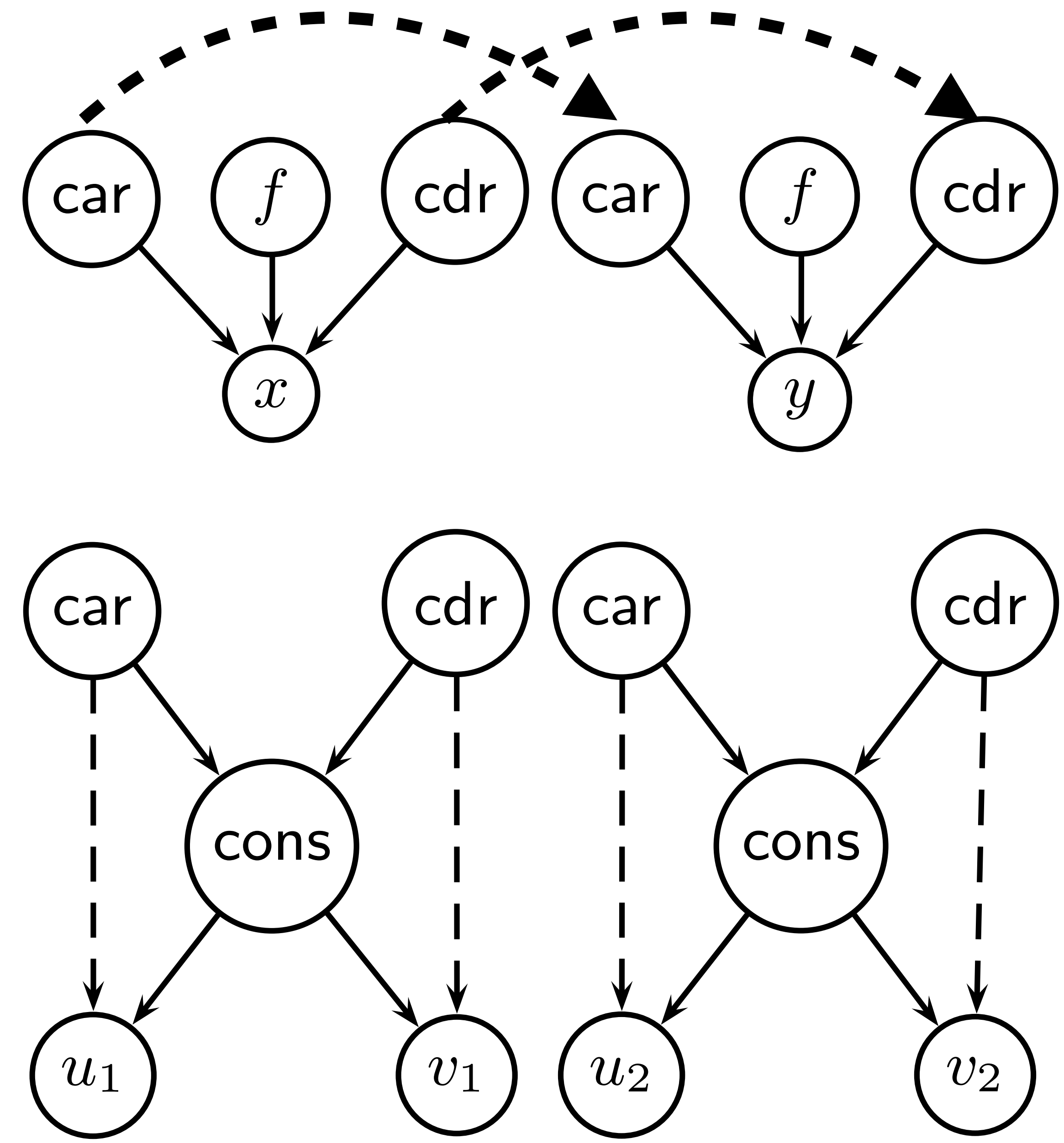
1. $car(x) = car(y)$
2. $cdr(x) = cdr(y)$



Combining T_E and T_{cons} - Example

Step 3: $\text{MERGE } s_i \ t_i$

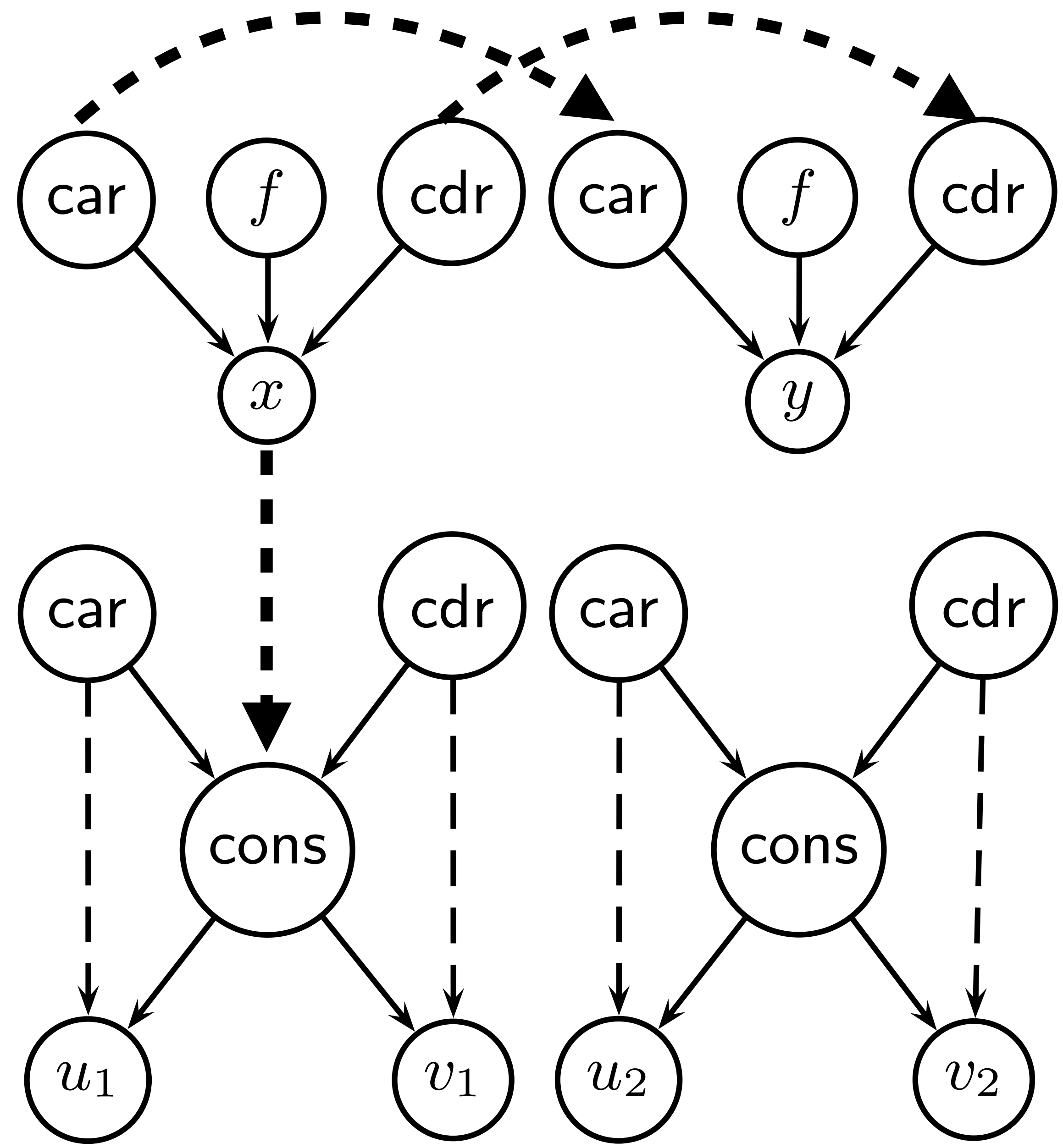
1. $car(x) = car(y)$
2. $cdr(x) = cdr(y)$
3. $x = cons(u_1, v_1)$



Combining T_E and T_{cons} - Example

Step 3: $\mathrm{MERGE}\ s_i\ t_i$

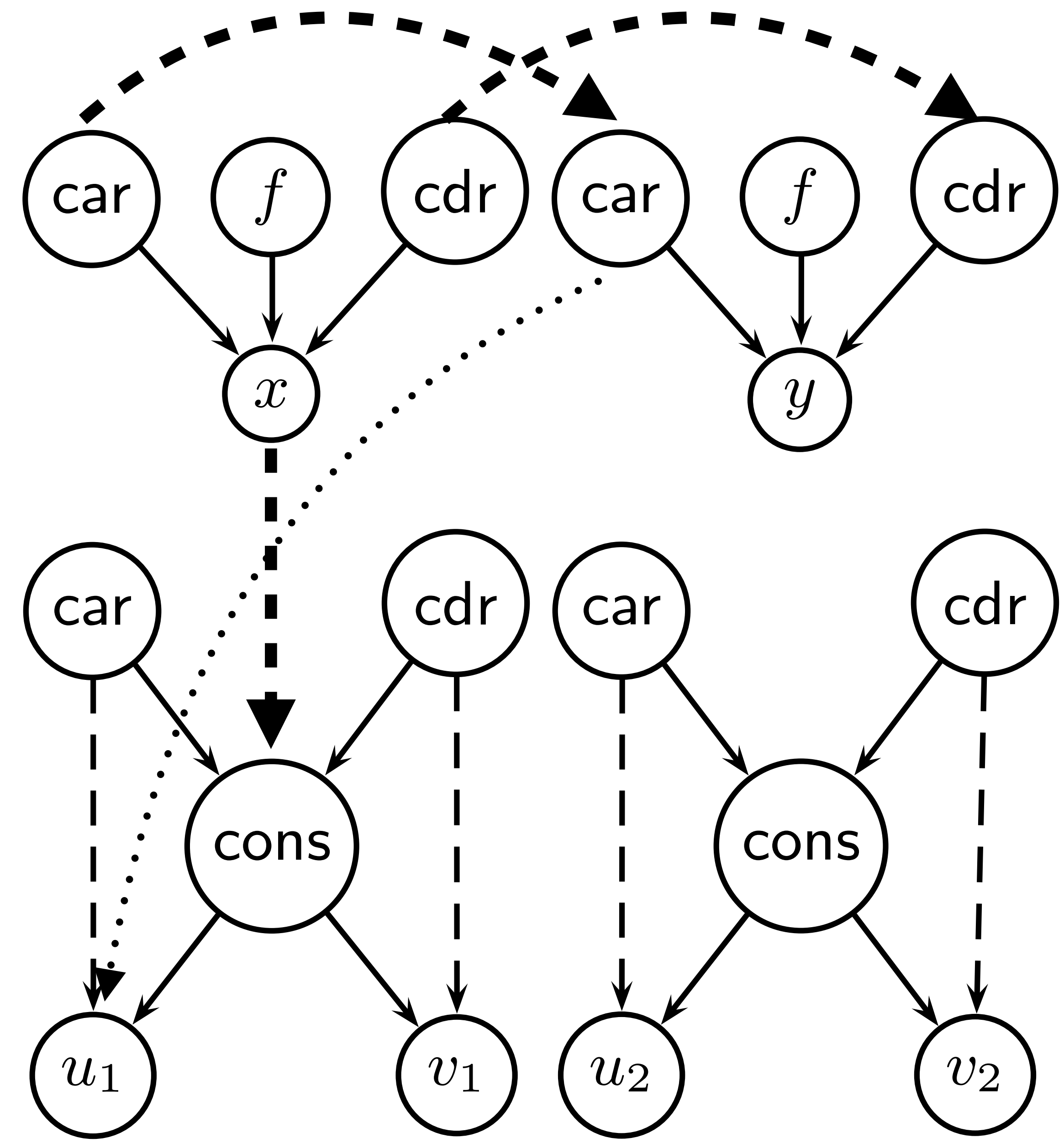
1. $car(x) = car(y)$
2. $cdr(x) = cdr(y)$
3. $x = cons(u_1,\ v_1)$



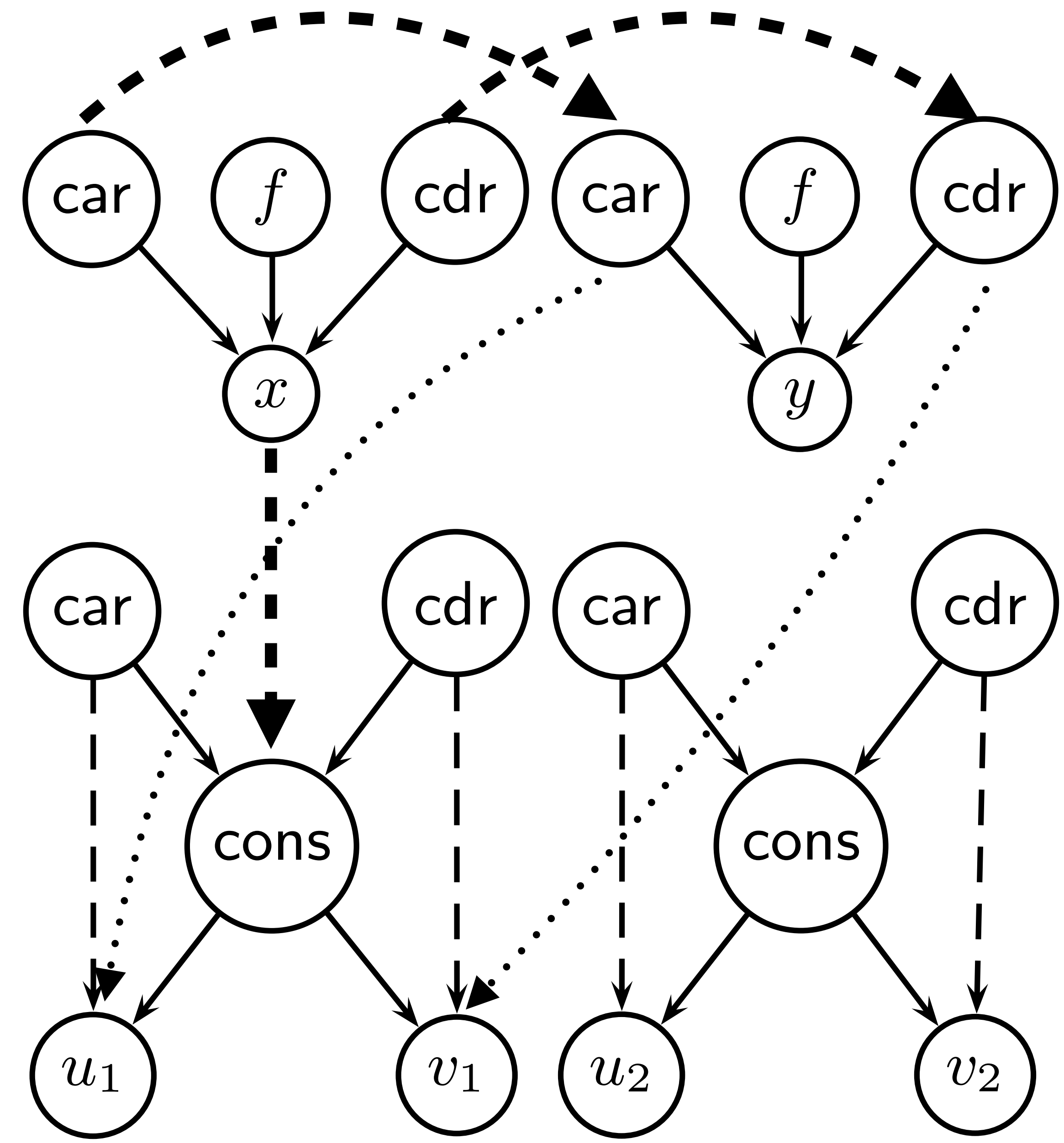
Combining T_E and T_{cons} - Example

Step 3: $\mathrm{MERGE}\ s_i\ t_i$

1. $car(x) = car(y)$
2. $cdr(x) = cdr(y)$
3. $x = cons(u_1, v_1)$



Combining T_E and T_{cons} - Example



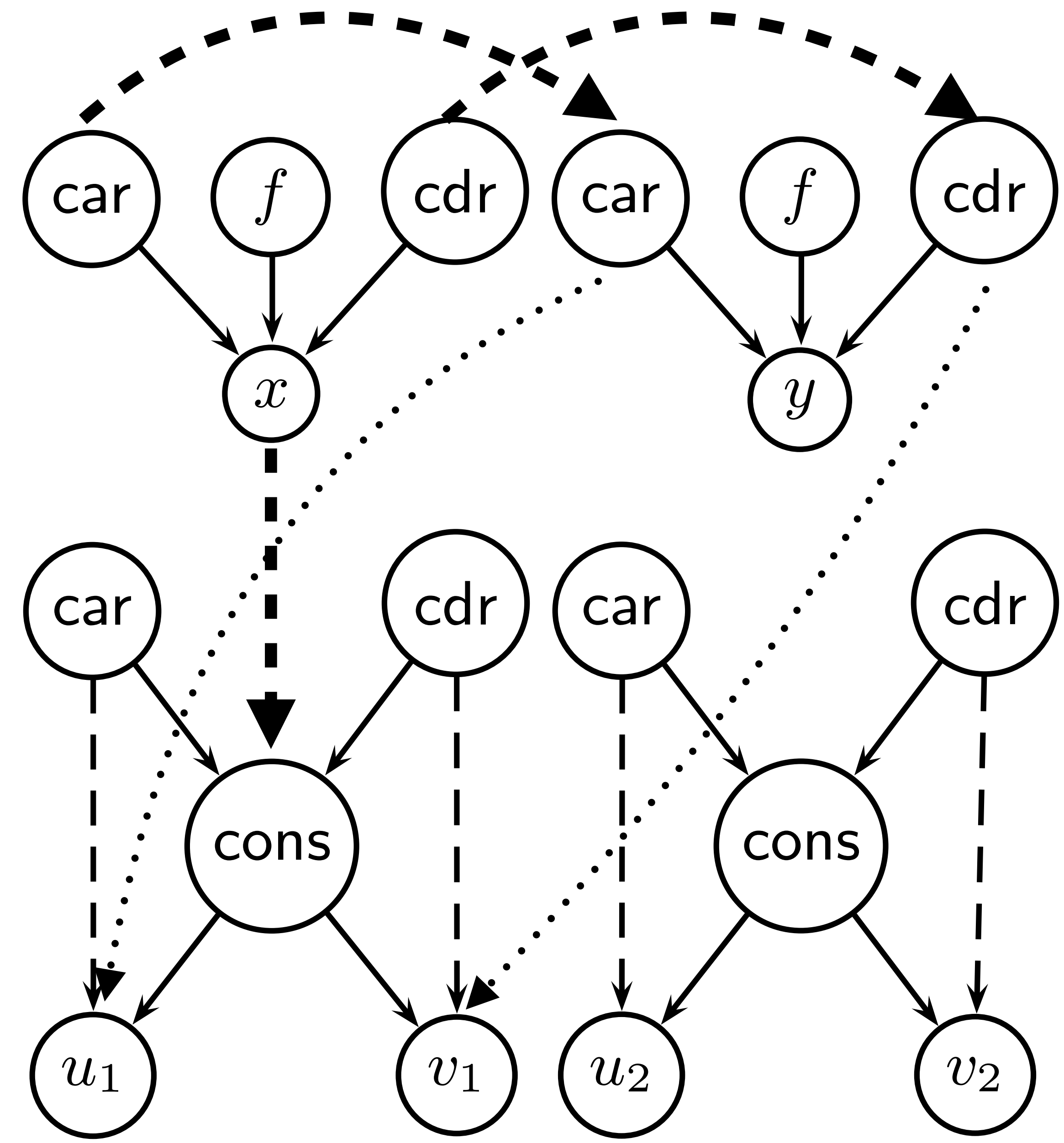
Step 3: $\mathrm{MERGE}\ s_i\ t_i$

1. $car(x) = car(y)$
2. $cdr(x) = cdr(y)$
3. $x = cons(u_1,\ v_1)$

Combining T_E and T_{cons} - Example

Step 3: $\text{MERGE } s_i \ t_i$

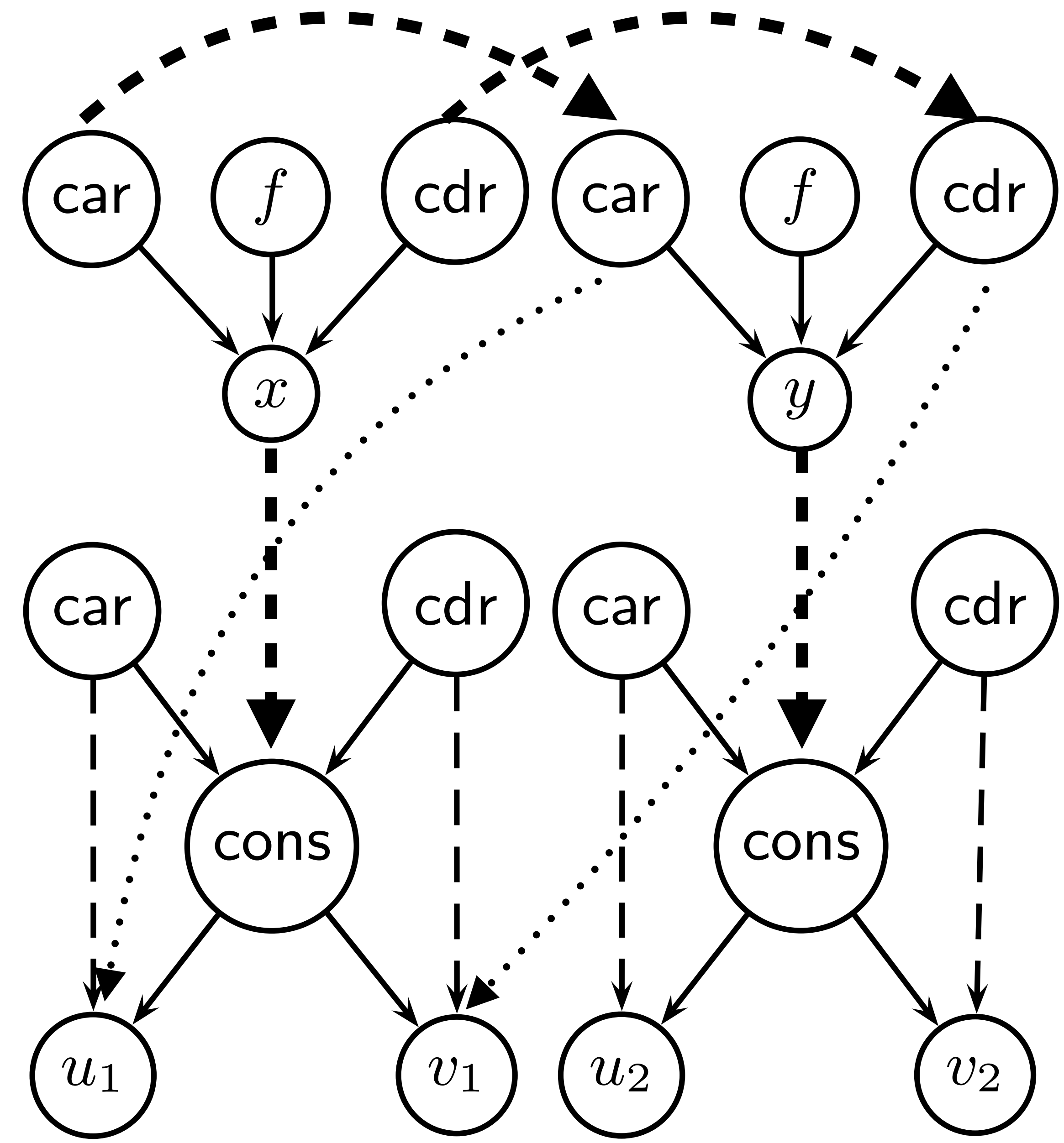
1. $car(x) = car(y)$
2. $cdr(x) = cdr(y)$
3. $x = cons(u_1, v_1)$
4. $y = cons(u_2, v_2)$



Combining T_E and T_{cons} - Example

Step 3: $\mathrm{MERGE}\ s_i\ t_i$

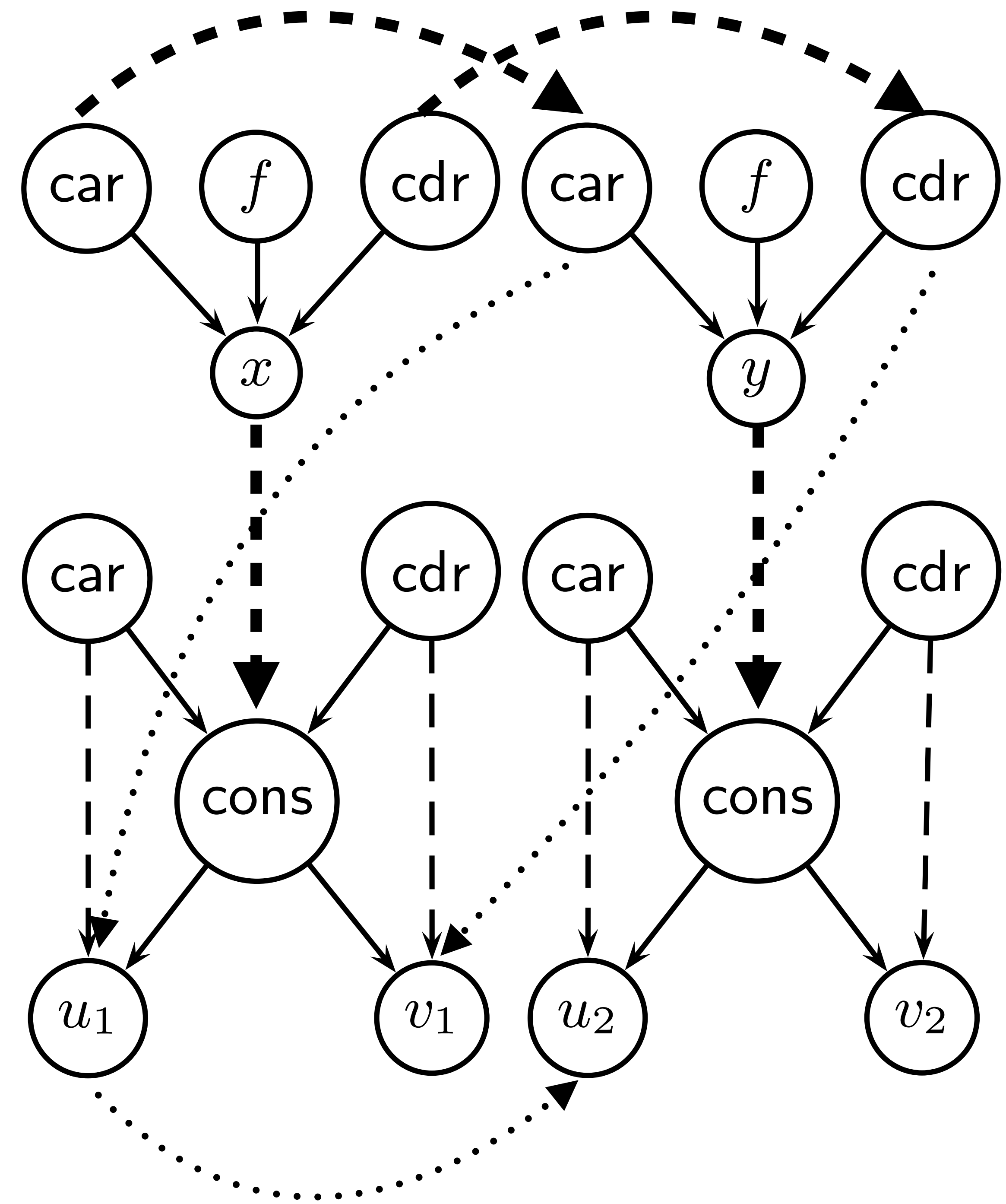
1. $car(x) = car(y)$
2. $cdr(x) = cdr(y)$
3. $x = cons(u_1, v_1)$
4. $y = cons(u_2, v_2)$



Combining T_E and T_{cons} - Example

Step 3: $\text{MERGE } s_i \ t_i$

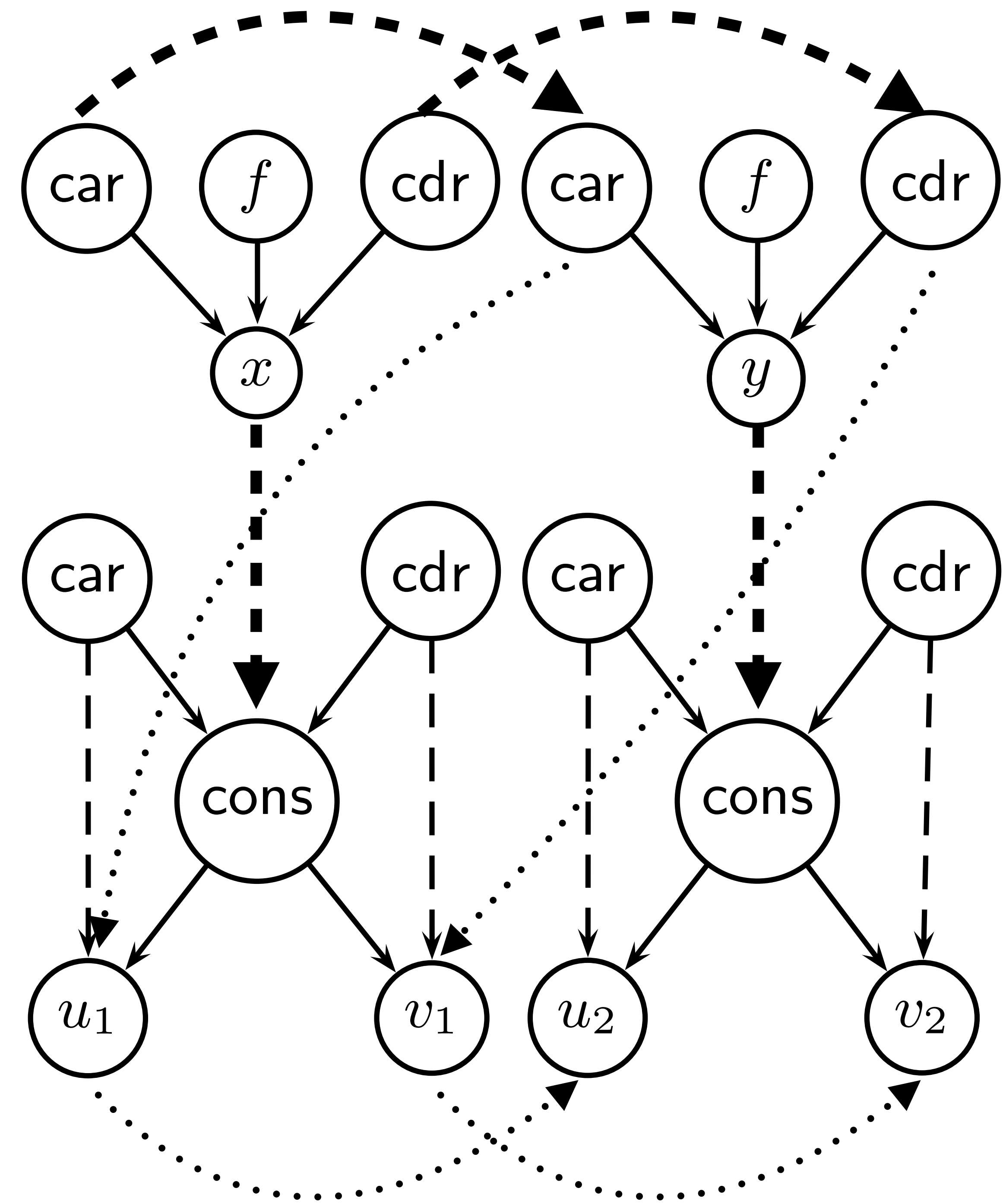
1. $car(x) = car(y)$
2. $cdr(x) = cdr(y)$
3. $x = cons(u_1, v_1)$
4. $y = cons(u_2, v_2)$



Combining T_E and T_{cons} - Example

Step 3: $\text{MERGE}\ s_i\ t_i$

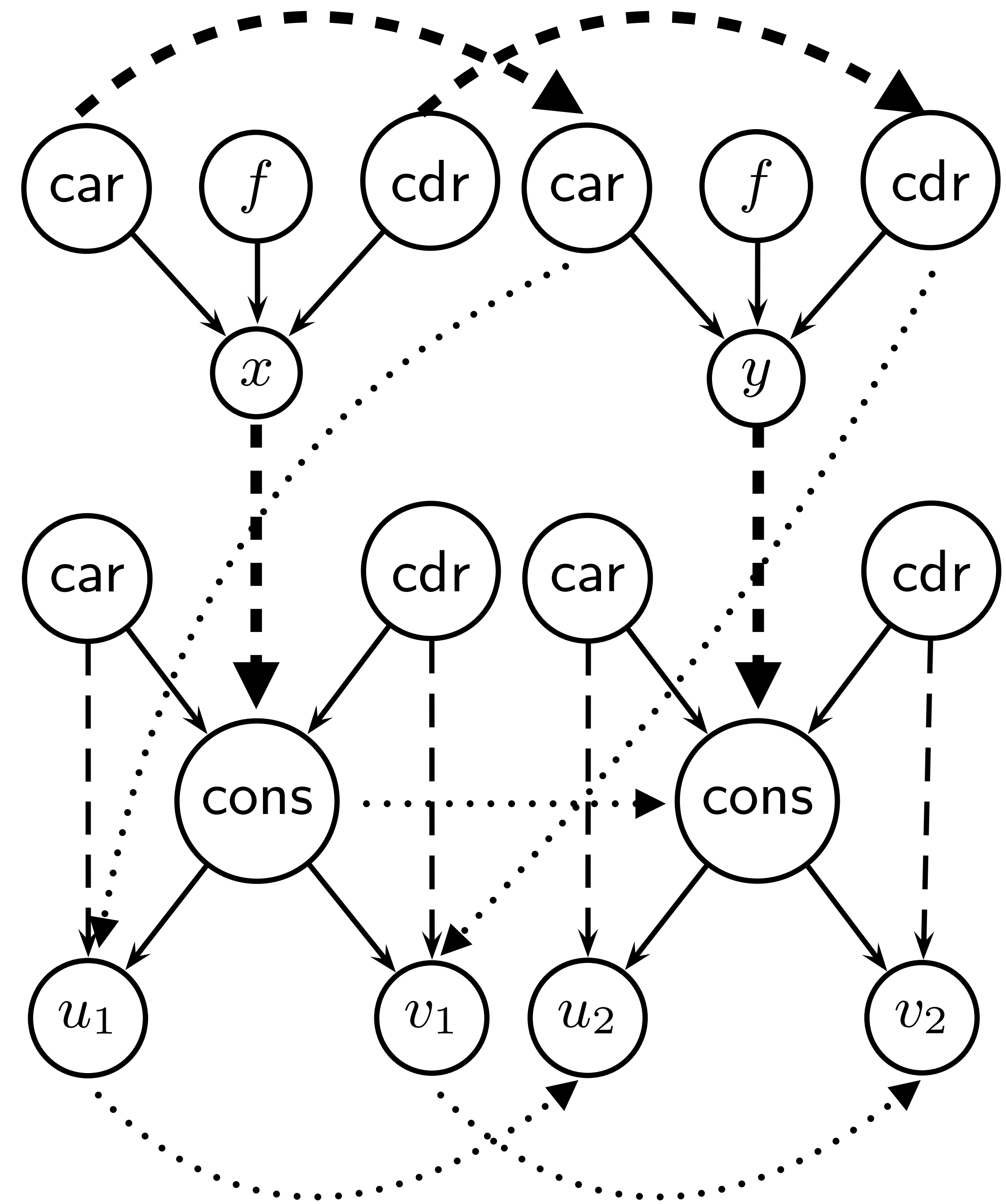
1. $car(x) = car(y)$
2. $cdr(x) = cdr(y)$
3. $x = cons(u_1, v_1)$
4. $y = cons(u_2, v_2)$



Combining T_E and T_{cons} - Example

Step 3: $\mathrm{MERGE}\ s_i\ t_i$

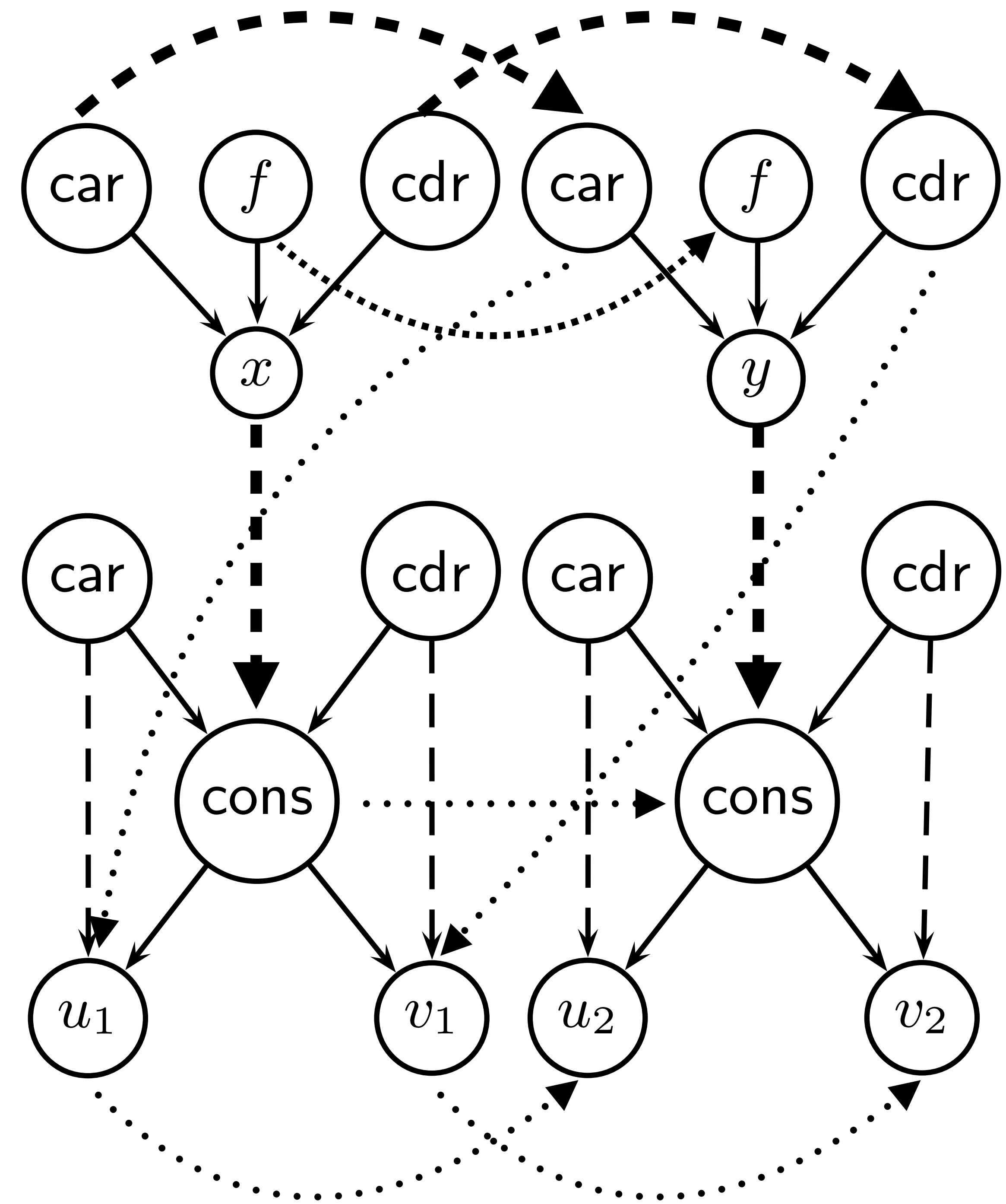
1. $car(x) = car(y)$
2. $cdr(x) = cdr(y)$
3. $x = cons(u_1,\ v_1)$
4. $y = cons(u_2,\ v_2)$



Combining T_E and T_{cons} - Example

Step 3: $\text{MERGE } s_i\ t_i$

1. $car(x) = car(y)$
2. $cdr(x) = cdr(y)$
3. $x = cons(u_1, v_1)$
4. $y = cons(u_2, v_2)$

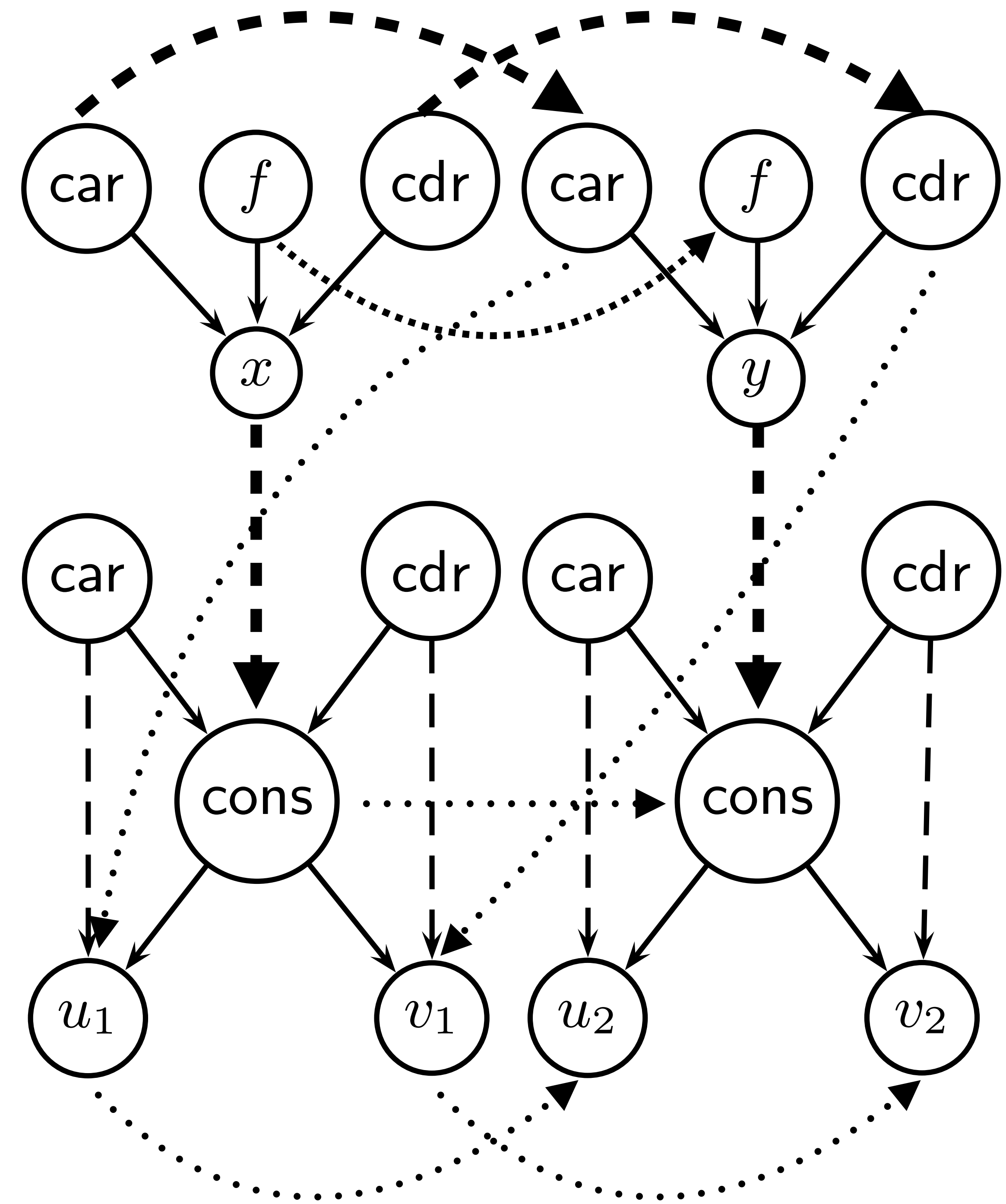


Combining T_E and T_{cons} - Example

Step 3: $\mathrm{MERGE}\ s_i\ t_i$

1. $car(x) = car(y)$
2. $cdr(x) = cdr(y)$
3. $x = cons(u_1, v_1)$
4. $y = cons(u_2, v_2)$

$(T_{cons} \cup T_E)$-unsatisfiable



Exercise

- Apply the decision procedure for $Tcons$ to the following $Tcons$-formulas. Please write down the call sequence to the MERGE procedure, draw the final DAG, and draw the final DAG.

  - $car(x) = y \wedge cdr(x) = z \wedge x \neq cons(y,z)$

  - $\neg atom(x) \wedge car(x) = y \wedge cdr(x) = z \wedge x \neq cons(y,z)$

Hint: Apply preprocessing to the formulae if it is necessary.

Arrays

Theory of Arrays - T_A

$$\Sigma_{\mathrm{A}} : \{\,\cdot[\cdot],\ \cdot\langle\cdot \triangleleft \cdot\rangle,\ =\}$$

- $a[i]$: a binary function; $a[i]$ represents the value of array a at position i;

- $a\langle i \triangleleft v\rangle$: a ternary function; $a\langle i \triangleleft v\rangle$ represents the modified array a in which position i has value v;

- $=$: a binary predicate

Axioms of T_A

- Axioms of (reflexivity), (symmetry), and (transitivity) of T_E
- $\forall a, i, j.\ i = j \rightarrow a[i] = a[j]$ (array congruence)
- $\forall a, v, i, j.\ i = j \rightarrow a\langle i \triangleleft v\rangle[j] = v$ (read-over-write 1)
- $\forall a, v, i, j.\ i \neq j \rightarrow a\langle i \triangleleft v\rangle[j] = a[j]$ (read-over-write 2)

Decision Procedure

- Based on a reduction to T_E-satisfiability via applications of the (read-over-write) axioms

- If the formula does not contain any write terms, then the read terms can be viewed as uninterpreted function terms

- Otherwise, any write term must occur in the context of a read

Decision Procedure - Step 1

If F does not contain any write terms $a\langle i \triangleleft v\rangle$, perform the following steps.

1. Associate each array variable a with a fresh function symbol f_a, and replace each read term $a[i]$ with $f_a(i)$
2. Decide and return the T_E-satisfiability of the resulting formula

Decision Procedure - Step 2

Select some read-over-write term $a\langle i \triangleleft v\rangle[j]$, and split on two cases:

1. According to (read-over-write 1), replace

   $$F[a\langle i \triangleleft v\rangle[j]] \text{ with } F_1\text{: } F[v] \wedge i = j$$

   and recurse on F_1. If F_1 is found to be T_A-satisfiable, return satisfiable

2. According to (read-over-write 2), replace

   $$F[a\langle i \triangleleft v\rangle[j]] \text{ with } F_2\text{: } F[a[j]] \wedge i \neq j$$

   and recurse on F_2. If F_2 is found to be T_A-satisfiable, return satisfiable

If both F_1 and F_2 are found to be T_A-unsatisfiable, return unsatisfiable

Example of T_A

$$F : i_1 = j \wedge i_1 \neq i_2 \wedge a[j] = v_1 \wedge a\langle i_1 \triangleleft v_1 \rangle \langle i_2 \triangleleft v_2 \rangle [j] \neq a[j]$$

- First case:
  - F_1: $i_2 = j \wedge i_1 = j \wedge i_1 \neq i_2 \wedge a[j] = v_1 \wedge v_2 \neq a[j]$
  - F_1': $i_2 = j \wedge i_1 = j \wedge i_1 \neq i_2 \wedge f_a(j) = v_1 \wedge v_2 \neq f_a(j)$
  - F_1 is T_A-unsatisfiable

Example of T_A

$$F : i_1 = j \wedge i_1 \neq i_2 \wedge a[j] = v_1 \wedge a\langle i_1 \triangleleft v_1 \rangle \langle i_2 \triangleleft v_2 \rangle [j] \neq a[j]$$

- Second case:
  - F_2: $i_2 \neq j \wedge i_1 = j \wedge i_1 \neq i_2 \wedge a[j] = v_1 \wedge a\langle i_1 \triangleleft v_1 \rangle [j] \neq a[j]$
    - F_3: $i_1 = j \wedge i_2 \neq j \wedge i_1 = j \wedge i_1 \neq i_2 \wedge a[j] = v_1 \wedge v_1 \neq a[j]$
    - F_4: $i_1 \neq j \wedge i_2 \neq j \wedge i_1 = j \wedge i_1 \neq i_2 \wedge a[j] = v_1 \wedge a[j] \neq a[j]$
  - F_2 is T_A-unsatisfiable

Soundness and Completeness

Theorem (Sound & Complete). Given quantifier-free conjunctive Σ_A-formula F, the decision procedure returns satisfiable iff F is T_A-satisfiable; otherwise, it returns unsatisfiable

Complexity

Theorem (Complexity). T_A-satisfiability of quantifier-free conjunctive Σ_A-formula is NP-complete

Exercise

- Apply the decision procedure for quantifier-free T_A to the following Σ_{A}-formulas.

  - $a\langle i \triangleleft e\rangle[j] = e \land i \neq j$

  - $a\langle i \triangleleft e\rangle\langle j \triangleleft f\rangle[k] = g \land j \neq k \land i = j \land a[k] \neq g$

Summary

- Congruence closure algorithm
  - relations, equivalence relations, congruence relations, partitions, quotients, classes, closures
- DAG-based implementation
  - union-find, merge
- Recursive data structures
  - T_{cons}
- Arrays