The Calculus of Computation:
Decision Procedures with Applications to Verification

by
Aaron Bradley
Zohar Manna

Springer 2007

9. Quantifier-free Equality and Data Structures

The Theory of Equality T_E

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

uninterpreted symbols:

- constants $a, b, c, \ldots$
- functions $f, g, h, \ldots$
- predicates $p, q, r, \ldots$

Example:

$x = y \ \wedge \ f(x) \neq f(y)$ T_E-unsatisfiable

$f(x) = f(y) \ \wedge \ x \neq y$ T_E-unsatisfiable

$f(f(f(a))) = a \ \wedge \ f(f(f(f(f(a))))) = a \ \wedge \ f(a) \neq a$ T_E-unsatisfiable

Axioms of T_E

1. $\forall x.\ x = x$ (reflexivity)
2. $\forall x, y.\ x = y \ \rightarrow \ y = x$ (symmetry)
3. $\forall x, y, z.\ x = y \ \wedge \ y = z \ \rightarrow \ x = z$ (transitivity)

define $=$ to be an equivalence relation.

Axiom schema

4. for each positive integer n and n-ary function symbol f,

$$\forall x_1, \ldots, x_n, y_1, \ldots, y_n.\ \bigwedge_i x_i = y_i \rightarrow \ f(x_1, \ldots, x_n) = f(y_1, \ldots, y_n)$$ (congruence)

For example,

$$\forall x, y.\ x = y \ \rightarrow \ f(x) = f(y)$$

Then

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

is T_E-valid.

Axiom schema

5. for each positive integer n and n-ary predicate symbol p,

$$\forall x_1, \ldots, x_n, y_1, \ldots, y_n. \bigwedge_i x_i = y_i \rightarrow$$
$$(p(x_1, \ldots, x_n) \leftrightarrow p(y_1, \ldots, y_n)) \qquad \text{(equivalence)}$$

Thus,

$$x = y \rightarrow (p(x) \leftrightarrow p(y))$$

is T_E-valid.

We discuss T_E-formulae without predicates

For example, for Σ_E-formula

$$F: \ p(x) \ \wedge \ q(x,y) \ \wedge \ q(y,z) \ \rightarrow \ \neg q(x,z)$$

introduce fresh constant $\bullet$ and fresh functions f_p and f_g, and transform F to

$$G: \ f_p(x) = \bullet \ \wedge \ f_q(x,y) = \bullet \ \wedge \ f_q(y,z) = \bullet \ \rightarrow \ f_q(x,z) \neq \bullet \ .$$

Equivalence and Congruence Relations: Basics

Binary relation R over set S

- is an equivalence relation if
  - reflexive: $\forall s \in S.\ sRs$;
  - symmetric: $\forall s_1, s_2 \in S.\ s_1 R s_2 \ \rightarrow\ s_2 R s_1$;
  - transitive: $\forall s_1, s_2, s_3 \in S.\ s_1 R s_2 \ \wedge\ s_2 R s_3 \ \rightarrow\ s_1 R s_3$.

Example:
Define the binary relation $\equiv_2$ over the set $\mathbb{Z}$ of integers

$$m \equiv_2 n \quad \text{iff} \quad (m \bmod 2) = (n \bmod 2)$$

That is, $m, n \in \mathbb{Z}$ are related iff they are both even or both odd.
$\equiv_2$ is an equivalence relation

- is a congruence relation if in addition

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

Classes

For $\left\{ \begin{array}{l} \text{equivalence} \\ \text{congruence} \end{array} \right\}$ relation R over set S,

The $\left\{ \begin{array}{l} \underline{\text{equivalence}} \\ \underline{\text{congruence}} \end{array} \right\}$ $\underline{\text{class}}$ of $s \in S$ under R is

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

Example:

The equivalence class of 3 under $\equiv_2$ over $\mathbb{Z}$ is

$$[3]_{\equiv_2} = \{n \in \mathbb{Z} \ : \ n \text{ is odd}\} \ .$$

Partitions

A $\underline{\text{partition}}$ P of S is a set of subsets of S that is

- $\underline{\text{total}}$ $\left(\bigcup_{S' \in P} S' \right) = S$
- $\underline{\text{disjoint}}$ $\forall S_1, S_2 \in P.\ S_1 \cap S_2 = \emptyset$

Quotient

The quotient S/R of S by $\left\{ \begin{array}{c} \text{equivalence} \\ \text{congruence} \end{array} \right\}$relation R is the set of $\left\{ \begin{array}{c} \text{equivalence} \\ \text{congruence} \end{array} \right\}$classes

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

It is a partition

Example: The quotient $\mathbb{Z}/\equiv_2$ is a partition of $\mathbb{Z}$. The set of equivalence classes

$$\{\{n \in \mathbb{Z} \;:\; n \text{ is odd}\},\; \{n \in \mathbb{Z} \;:\; n \text{ is even}\}\}$$

Note duality between relations and classes

Refinements

Two binary relations R_1 and R_2 over set S.
R_1 is refinement of R_2, $R_1 \prec R_2$, if

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

R_1 refines R_2.

Examples:

- For $S = \{a, b\}$,
  $R_1 : \{aR_1 b\}$ $\quad$ $R_2 : \{aR_2 b,\ bR_2 b\}$
  Then $R_1 \prec R_2$
- For set S,
  R_1 induced by the partition $P_1 : \{\{s\}\ :\ s \in S\}$
  R_2 induced by the partition $P_2 : \{S\}$
  Then $R_1 \prec R_2$.
- For set $\mathbb{Z}$
  $R_1 : \{xR_1 y\ :\ x \bmod 2 = y \bmod 2\}$
  $R_2 : \{xR_2 y\ :\ x \bmod 4 = y \bmod 4\}$
  Then $R_2 \prec R_1$.

Closures

Given binary relation R over S.

The equivalence closure R^E of R is the equivalence relation s.t.

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

That is, R^E is the "smallest" equivalence relation that "covers" R.

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

Hence,

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

Similarly, the congruence closure R^C of R is the "smallest" congruence relation that "covers" R.

Congruence Closure Algorithm

Given Σ_E-formula

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

decide if F is Σ_E-satisfiable.

Definition: For Σ_E-formula F,
the subterm set S_F of F is the set that contains precisely the subterms of F.

Example: The subterm set of

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

is

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

The Algorithm

Given Σ_E-formula F

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

with subterm set S_F, 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 $i \in \{m+1, \ldots, n\}$, $s_i \not\sim t_i$.

Such congruence relation $\sim$ defines T_E-interpretation $I : (D_I, \alpha_I)$ of F. D_I consists of $|S_F/\sim|$ elements, one for each congruence class of S_F under $\sim$.

Instead of writing $I \models F$ for this T_E-interpretation, we abbreviate

$$\sim \models F$$

The goal of the algorithm is to construct the congruence relation of S_F, or to prove that no congruence relation exists.

$$F: \underbrace{s_1 = t_1 \ \wedge \ \cdots \ \wedge \ s_m = t_m}_{\text{generate congruence closure}} \wedge \underbrace{s_{m+1} \neq t_{m+1} \ \wedge \ \cdots \ \wedge \ s_n \neq t_n}_{\text{search for contradiction}}$$

The algorithm performs the following steps:

1. Construct the congruence closure $\sim$ of

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

over the subterm set S_F. Then

$$\sim \models \ s_1 = t_1 \ \wedge \ \cdots \ \wedge \ s_m = t_m \ .$$

2. If for any $i \in \{m+1, \ldots, n\}$, $s_i \sim t_i$, return unsatisfiable.
3. Otherwise, $\sim \not\models F$, so return satisfiable.

How do we actually construct the congruence closure in Step 1?

Initially, begin with the finest congruence relation $\sim_0$ given by the partition

$$\{\{s\} \ : \ s \in S_F\} \ .$$

That is, let each term of S_F be its own congruence class.

Then, for each $i \in \{1, \ldots, m\}$, impose $s_i = t_i$ by merging the congruence classes

$$[s_i]_{\sim_{i-1}} \quad \text{and} \quad [t_i]_{\sim_{i-1}}$$

to form a new congruence relation $\sim_i$. To accomplish this merging,

- form the union of $[s_i]_{\sim_{i-1}}$ and $[t_i]_{\sim_{i-1}}$
- propagate any new congruences that arise within this union.

The new relation $\sim_i$ is a congruence relation in which $s_i \sim t_i$.

Example: Given Σ_E-formula

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

Construct initial partition by letting each member of the subterm set S_F be its own class:

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

According to the first literal $f(a,b) = a$, merge

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

to form partition

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

According to the (congruence) axiom,

$f(a,b) \sim a, \ b \sim b$ implies $f(f(a,b),b) \sim f(a,b)$,

resulting in the new partition

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

This partition represents the congruence closure of S_F. Now, is it the case that

4. $\{\{a, f(a,b), f(f(a,b),b)\}, \ \{b\}\} \ \models \ F$?

No, as $f(f(a,b),b) \sim a$ but F asserts that $f(f(a,b),b) \neq a$. Hence, F is T_E-unsatisfiable.

Example: Given Σ_E-formula

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

From the subterm set S_F, the initial partition is

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

where, for example, $f^3(a)$ abbreviates $f(f(f(a)))$.

According to the literal $f^3(a) = a$, merge

$$\{f^3(a)\} \quad \text{and} \quad \{a\} \; .$$

From the union,

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

deduce the following congruence propagations:

$$f^3(a) \sim a \; \Rightarrow \; f(f^3(a)) \sim f(a) \quad \text{i.e.} \quad f^4(a) \sim f(a)$$

and

$$f^4(a) \sim f(a) \; \Rightarrow \; f(f^4(a)) \sim f(f(a)) \; \text{ i.e. } \; f^5(a) \sim f^2(a)$$

Thus, the final partition for this iteration is the following:

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

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

From the second literal, $f^5(a) = a$, merge

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

to form the partition

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

Propagating the congruence

$f^3(a) \sim f^2(a) \ \Rightarrow\ f(f^3(a)) \sim f(f^2(a))$ i.e. $f^4(a) \sim f^3(a)$

yields the partition

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

which represents the congruence closure in which all of S_F are equal. Now,

6. $\{\{a, f(a), f^2(a), f^3(a), f^4(a), f^5(a)\}\} \models F$?

No, as $f(a) \sim a$, but F asserts that $f(a) \neq a$. Hence, F is T_E-unsatisfiable.

Example: Given Σ_E-formula

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

The subterm set S_F induces the following initial partition:

1. $\{\{x\}, \ \{y\}, \ \{f(x)\}, \ \{f(y)\}\}$.

Then $f(x) = f(y)$ indicates to merge

$\{f(x)\}$ and $\{f(y)\}$.

The union $\{f(x), f(y)\}$ does not yield any new congruences, so the final partition is

2. $\{\{x\}, \ \{y\}, \ \{f(x), f(y)\}\}$.

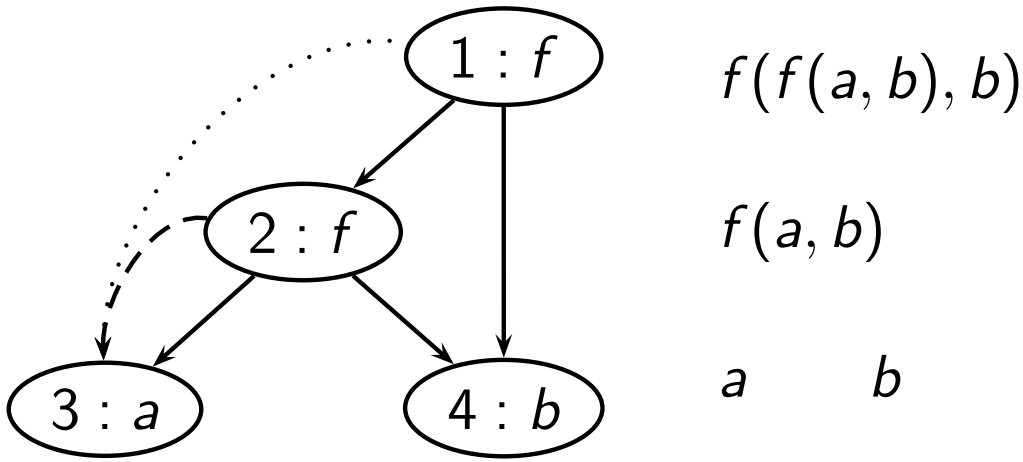
Does

3. $\{\{x\}, \ \{y\}, \ \{f(x), f(y)\}\} \models F$?

Yes, as $x \not\sim y$, agreeing with $x \neq y$. Hence, F is T_E-satisfiable.

Directed Acyclic Graph (DAG)

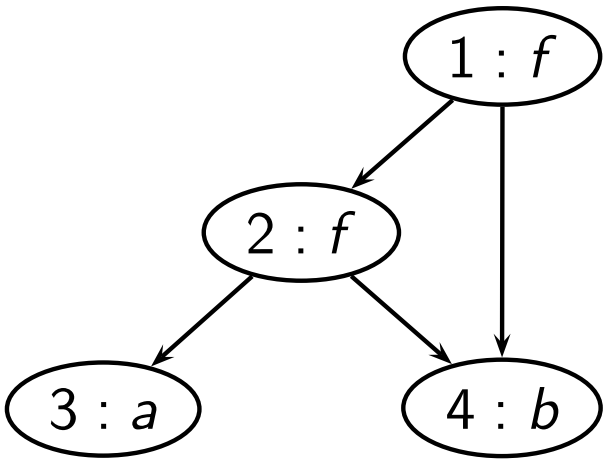
For Σ_E-formula F, graph-based data structure for representing the subterms of S_F (and congruence relation between them).



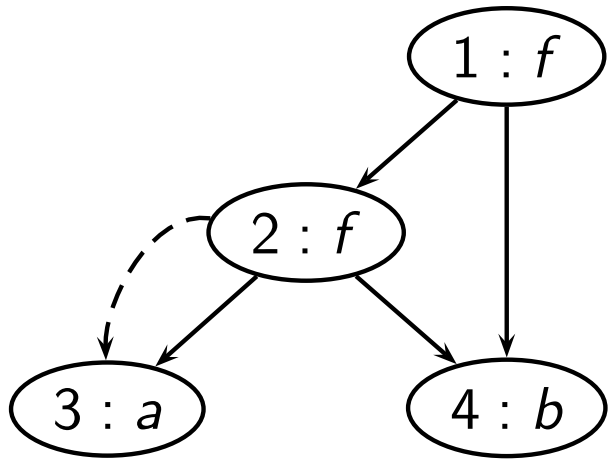
Efficient way for computing the congruence closure algorithm.

T_E-Satisfiability (Summary of idea)

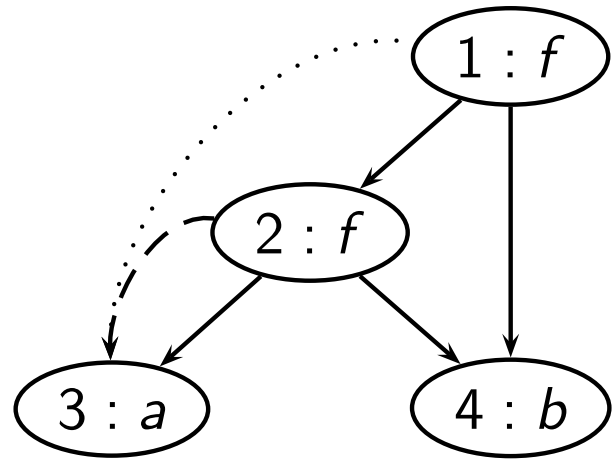
$$f(a,b) = a \ \wedge \ f(f(a,b),b) \neq a$$



Initial DAG



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



$f(a,b) \sim a, \ b \sim b \ \Rightarrow$
$f(f(a,b),b) \sim f(a,b)$
MERGE $f(f(a,b),b)$
$f(a,b)$

_ _ _ explicit equation ..... by congruence

$$\left.\begin{array}{r} \text{FIND } f(f(a,b),b) = a = \text{FIND } a \\ f(f(a,b),b) \neq a \end{array}\right\} \Rightarrow \textbf{Unsatisfiable}$$

DAG representation

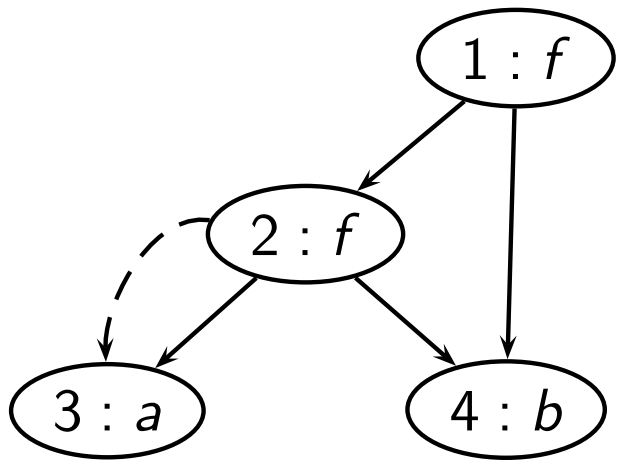
`type` **node** $=$ $\{$

`id`	:	**id** node's unique identification number
`fn`	:	**string** constant or function name
`args`	:	**id list** list of function arguments
`mutable find`	:	**id** the representative of the congruence class
`mutable ccpar`	:	**id set** if the node is the representative for its congruence class, then its `ccpar` (congruence closure parents) are all parents of nodes in its congruence class

$\}$

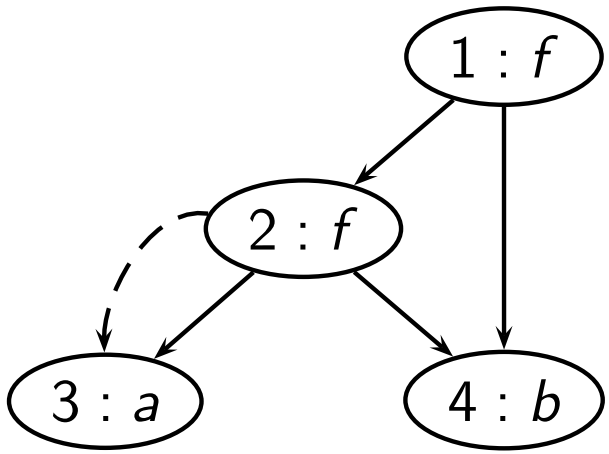
DAG Representation of node 2

```
type node = {
    id                  :  id       ... 2
    fn                  :  string   ... f
    args                :  idlist   ... [3, 4]
    mutable find        :  id       ... 3
    mutable ccpar       :  idset    ... ∅
  }
```



DAG Representation of node 3

```
type node = {
    id                  : id        ... 3
    fn                  : string    ... a
    args                : idlist    ... []
    mutable find        : id        ... 3
    mutable ccpar       : idset     ... {1, 2}
  }
```

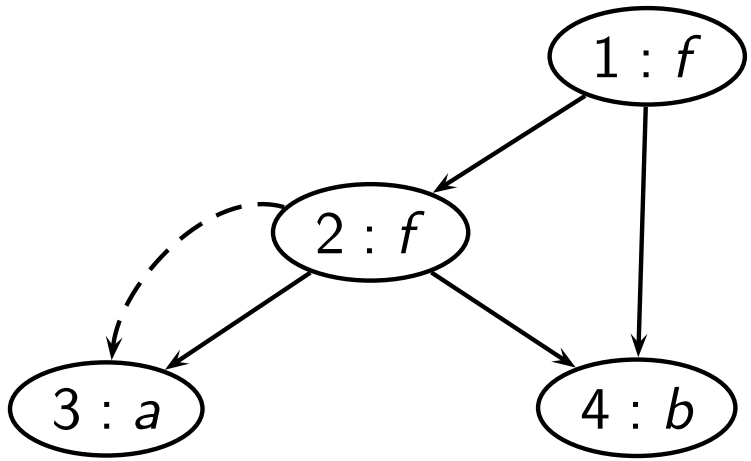


The Implementation

FIND function

returns the representative of node's congruence class

```
let rec FIND i =
  let n = NODE i in
  if n.find = i then i else FIND n.find
```



Example: FIND 2 = 3

FIND 3 = 3

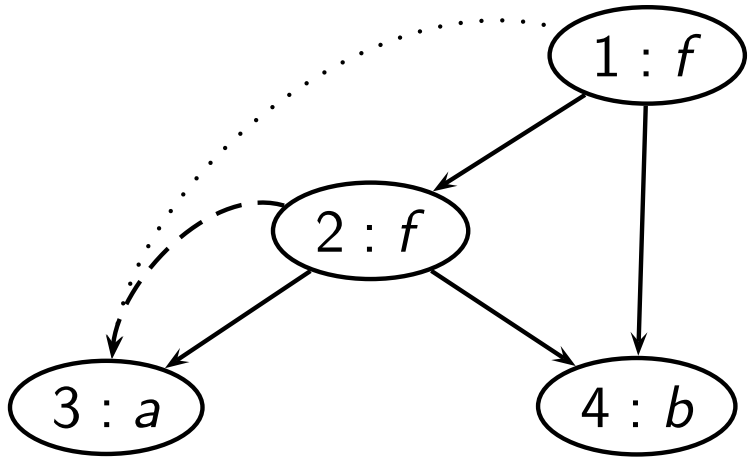
3 is the representative of 2.

UNION function

```
let UNION i1 i2 =
  let n1 = NODE (FIND i1) in
  let n2 = NODE (FIND i2) in
  n1.find ← n2.find;
  n2.ccpar ← n1.ccpar ∪ n2.ccpar;
  n1.ccpar ← ∅
```

n_2 is the representative of the union class

Example



UNION 1 2 $\quad n_1 = 1 \quad n_2 = 3$

$1.\texttt{find} \leftarrow 3$

$3.\texttt{ccpar} \leftarrow \{1, 2\}$

$1.\texttt{ccpar} \leftarrow \emptyset$

CCPAR function

Returns parents of all nodes in i's congruence class

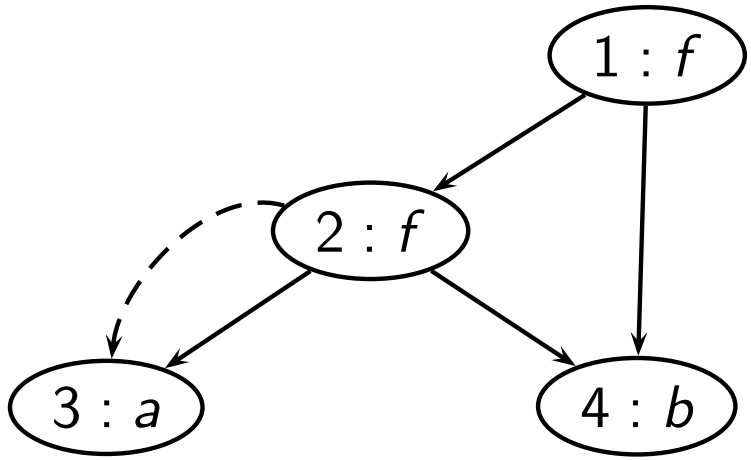
$$\texttt{let CCPAR } i = \\ \quad (\text{NODE } (\text{FIND } i)).\texttt{ccpar}$$

CONGRUENT predicate

Test whether i_1 and i_2 are congruent

$$\begin{array}{l}
\texttt{let CONGRUENT } i_1\ i_2 = \\
\quad \texttt{let } n_1 = \text{NODE } i_1 \texttt{ in} \\
\quad \texttt{let } n_2 = \text{NODE } i_2 \texttt{ in} \\
\quad n_1.\texttt{fn} = n_2.\texttt{fn} \\
\qquad \wedge\ |n_1.\texttt{args}| = |n_2.\texttt{args}| \\
\qquad \wedge\ \forall i \in \{1, \ldots, |n_1.\texttt{args}|\}.\ \text{FIND } n_1.\texttt{args}[i] = \text{FIND } n_2.\texttt{args}[i]
\end{array}$$

Example:



Are 1 and 2 congruent?

`fn` fields	— both f
# of arguments	— same
left arguments $f(a, b)$ and a	— both congruent to 3
right arguments b and b	— both 4 (congruent)

Therefore 1 and 2 are congruent.

MERGE function

```
let rec MERGE i1 i2 =
  if FIND i1 ≠ FIND i2 then begin
    let P_i1 = CCPAR i1 in
    let P_i2 = CCPAR i2 in
    UNION i1 i2;
    foreach t1, t2 ∈ P_i1 × P_i2 do
      if FIND t1 ≠ FIND t2 ∧ CONGRUENT t1 t2
      then MERGE t1 t2
    done
  end
```

P_{i_1} and P_{i_2} store the current values of CCPAR i_1 and CCPAR i_2.

Decision Procedure: T_E-satisfiability

Given Σ_E-formula

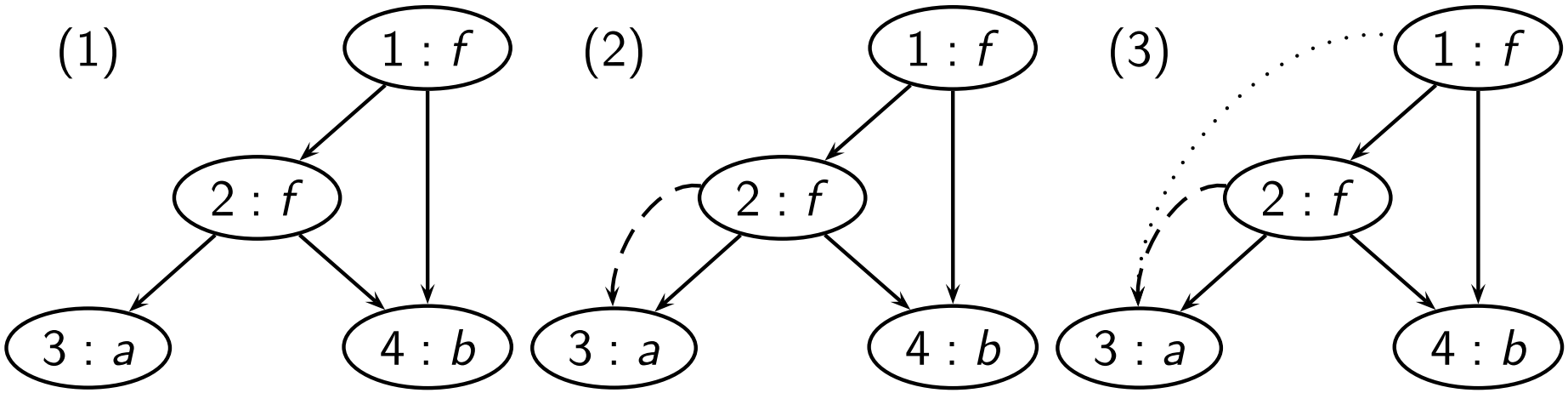
$$F: \; s_1 = t_1 \wedge \cdots \wedge s_m = t_m \wedge s_{m+1} \neq t_{m+1} \wedge \cdots \wedge s_n \neq t_n \;,$$

with subterm set S_F, perform the following steps:

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

Example 1: T_E-Satisfiability

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



Initial DAG

MERGE 2 3

UNION 2 3

$P_2 = \{1\}$

$P_3 = \{2\}$

CONGRUENT 1 2

MERGE 1 2

UNION 1 2

$P_1 = \{\}$

$P_2 = \{1,2\}$

FIND $f(f(a,b),b) = a =$ FIND $a \Rightarrow$ **Unsatisfiable**

Given Σ_E-formula

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

The subterm set is

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

resulting in the initial partition

$$(1) \ \{\{a\}, \ \{b\}, \ \{f(a,b)\}, \ \{f(f(a,b),b)\}\}$$

in which each term is its own congruence class. Fig (1).

Final partition

$$(2) \ \{\{a, f(a,b), f(f(a,b),b)\}, \ \{b\}\}$$

Note: dash edge _____ merge dictated by equalities in F
dotted edge ..... deduced merge

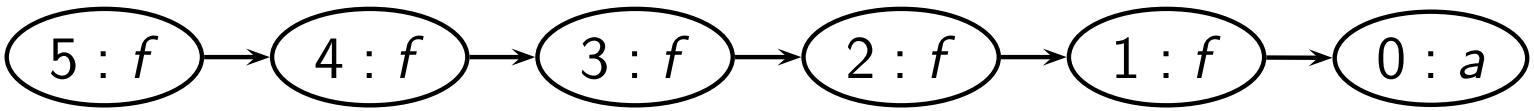
Does

$$(3) \ \{\{a, f(a,b), f(f(a,b),b)\}, \ \{b\}\} \ \models \ F \ ?$$

No, as $f(f(a,b),b) \sim a$, but F asserts that $f(f(a,b),b) \neq a$.
Hence, F is T_E-unsatisfiable.

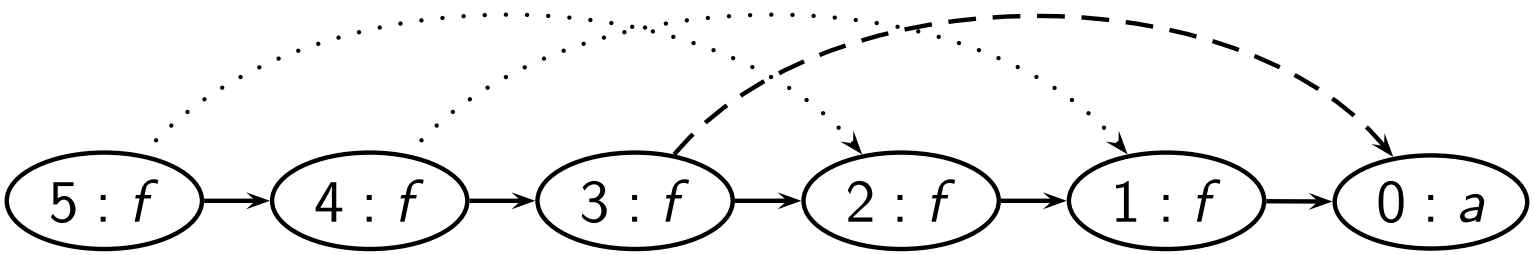
Example 2: T_E-Satisfiability

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



(1)

Initial DAG



(2)

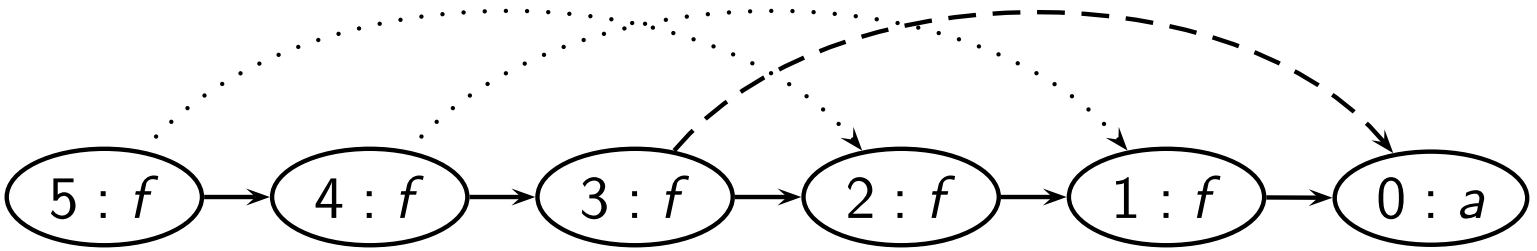
$f(f(f(a))) = a \ \Rightarrow \ \text{MERGE } 3\ 0 \quad P_3 = \{4\} \ \ P_0 = \{1\}$

$\Rightarrow \ \text{MERGE } 4\ 1 \quad P_4 = \{5\} \ \ P_1 = \{2\}$

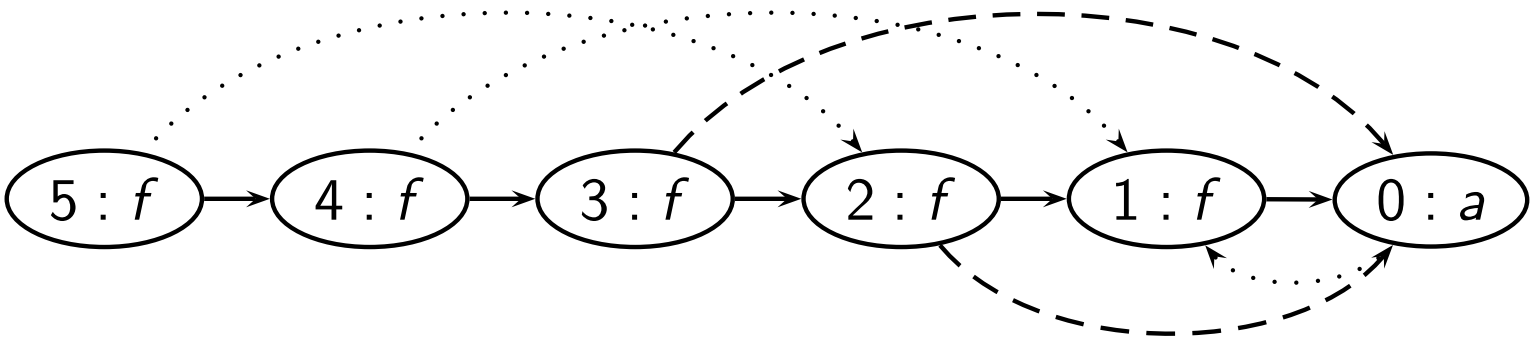
$\Rightarrow \ \text{MERGE } 5\ 2 \quad P_5 = \{\} \ \ P_2 = \{3\}$

Example 2: T_E-Satisfiability

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



(2)



(3)

$f(f(f(f(f(a))))) = a \quad \Rightarrow \ \text{MERGE } 5\ 0 \quad P_5 = \{3\} \quad P_0 = \{1, 4\}$

$\Rightarrow \ \text{MERGE } 3\ 1 \quad \text{STOP. Why?}$

$\text{FIND } f(a) = f(a) = \text{FIND } a \ \Rightarrow$ **Unsatisfiable**

Given Σ_E-formula

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

which induces the initial partition

1. $\{\{a\}, \{f(a)\}, \{f^2(a)\}, \{f^3(a)\}, \{f^4(a)\}, \{f^5(a)\}\}$.
   The equality $f^3(a) = a$ induces the partition
2. $\{\{a, f^3(a)\}, \{f(a), f^4(a)\}, \{f^2(a), f^5(a)\}\}$.
   The equality $f^5(a) = a$ induces the partition
3. $\{\{a, f(a), f^2(a), f^3(a), f^4(a), f^5(a)\}\}$.
   Now, does

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

   No, as $f(a) \sim a$, but F asserts that $f(a) \neq a$. Hence, F is T_E-unsatisfiable.

Theorem (Sound and Complete)

Quantifier-free conjunctive Σ_E-formula F is T_E-satisfiable iff the congruence closure algorithm returns satisfiable.

Recursive Data Structures

Quantifier-free Theory of Lists T_{cons}

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

- constructor cons : cons(a, b) list constructed by prepending a to b
- left projector car : car(cons(a, b)) $= a$
- right projector cdr : cdr(cons(a, b)) $= b$
- atom : unary predicate

Axioms of T_{cons}

- reflexivity, symmetry, transitivity
- congruence axioms:

$$\forall x_1, x_2, y_1, y_2.\ x_1 = x_2 \ \wedge\ y_1 = y_2 \ \rightarrow\ \text{cons}(x_1, y_1) = \text{cons}(x_2, y_2)$$
$$\forall x, y.\ x = y \ \rightarrow\ \text{car}(x) = \text{car}(y)$$
$$\forall x, y.\ x = y \ \rightarrow\ \text{cdr}(x) = \text{cdr}(y)$$

- equivalence axiom:

$$\forall x, y.\ x = y \ \rightarrow\ (\text{atom}(x) \ \leftrightarrow\ \text{atom}(y))$$

-

$$(A1)\ \forall x, y.\ \text{car}(\text{cons}(x, y)) = x \qquad \text{(left projection)}$$
$$(A2)\ \forall x, y.\ \text{cdr}(\text{cons}(x, y)) = y \qquad \text{(right projection)}$$
$$(A3)\ \forall x.\ \neg\text{atom}(x) \rightarrow \text{cons}(\text{car}(x), \text{cdr}(x)) = x \qquad \text{(construction)}$$
$$(A4)\ \forall x, y.\ \neg\text{atom}(\text{cons}(x, y)) \qquad \text{(atom)}$$

Simplifications

- Consider only quantifier-free conjunctive Σ_{cons}-formulae. Convert non-conjunctive formula to DNF and check each disjunct.
- $\neg\text{atom}(u_i)$ literals are removed:

  $$\boxed{\text{replace} \quad \neg\text{atom}(u_i) \quad \text{with} \quad u_i = \text{cons}(u_i^1, u_i^2)}$$

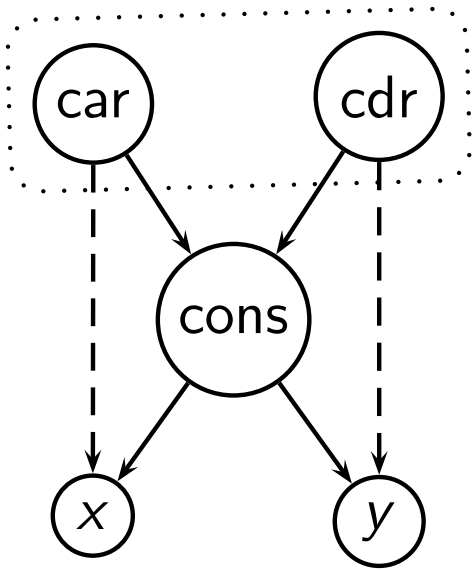
  by the (construction) axiom.
- Because of similarity to Σ_{E}, we sometimes combine $\Sigma_{\text{cons}} \cup \Sigma_{\text{E}}$.

Algorithm: T_{cons}-Satisfiability (the idea)

$$
\begin{aligned}
F: \quad & \underbrace{s_1 = t_1 \ \wedge \ \cdots \ \wedge \ s_m = t_m}_{\text{generate congruence closure}} \\
\wedge \quad & \underbrace{s_{m+1} \neq t_{m+1} \ \wedge \ \cdots \ \wedge \ s_n \neq t_n}_{\text{search for contradiction}} \\
\wedge \quad & \underbrace{\text{atom}(u_1) \ \wedge \ \cdots \ \wedge \ \text{atom}(u_\ell)}_{\text{search for contradiction}}
\end{aligned}
$$

where s_i, t_i, and u_i are T_{cons}-terms

Algorithm: T_{cons}-Satisfiability



1. Construct the initial DAG for S_F
2. for each node n with $n.\texttt{fn} = \text{cons}$
   - add car(n) and $\textsc{merge}$ car(n) $n.\texttt{args}[1]$
   - add cdr(n) and $\textsc{merge}$ cdr(n) $n.\texttt{args}[2]$

   by axioms (A1), (A2)
3. for $1 \leq i \leq m$, $\textsc{merge}\ s_i\ t_i$
4. for $m+1 \leq i \leq n$, if $\textsc{find}\ s_i = \textsc{find}\ t_i$, return **unsatisfiable**
5. for $1 \leq i \leq \ell$, if $\exists v.\ \textsc{find}\ v = \textsc{find}\ u_i\ \wedge\ v.\texttt{fn} = \text{cons}$, return **unsatisfiable**
6. Otherwise, return **satisfiable**

Example:

Given $(\Sigma_{\text{cons}} \cup \Sigma_{\text{E}})$-formula

$$F: \quad \begin{array}{c} \text{car}(x) = \text{car}(y) \ \wedge \ \text{cdr}(x) = \text{cdr}(y) \\ \wedge \ \neg\text{atom}(x) \ \wedge \ \neg\text{atom}(y) \ \wedge \ f(x) \neq f(y) \end{array}$$

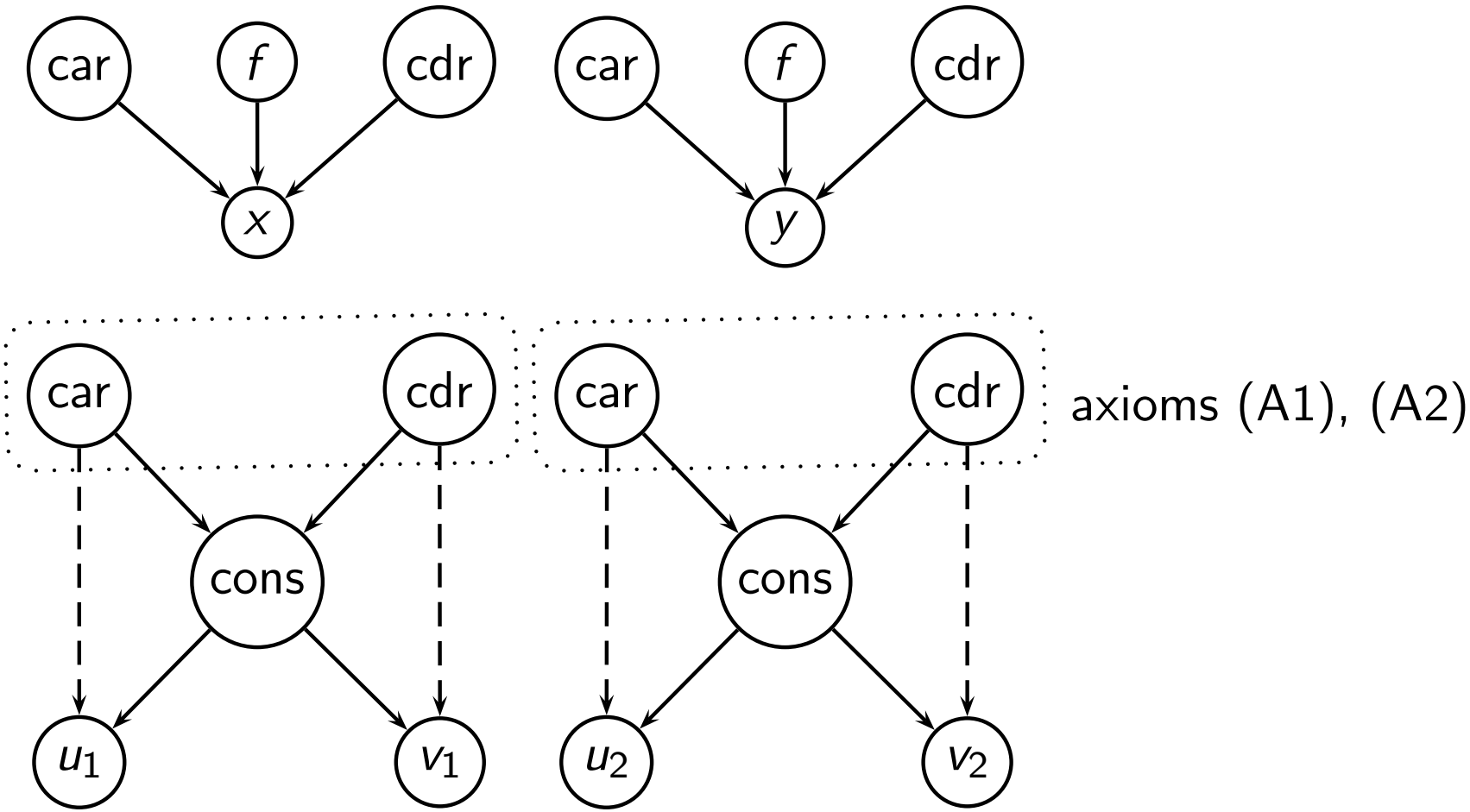
where the function symbol f is in Σ_{E}

$$F': \quad \begin{array}{lll} \text{car}(x) = \text{car}(y) & \wedge & (1) \\ \text{cdr}(x) = \text{cdr}(y) & \wedge & (2) \\ x = \text{cons}(u_1, v_1) & \wedge & (3) \\ y = \text{cons}(u_2, v_2) & \wedge & (4) \\ f(x) \neq f(y) & & (5) \end{array}$$

Recall the projection axioms:

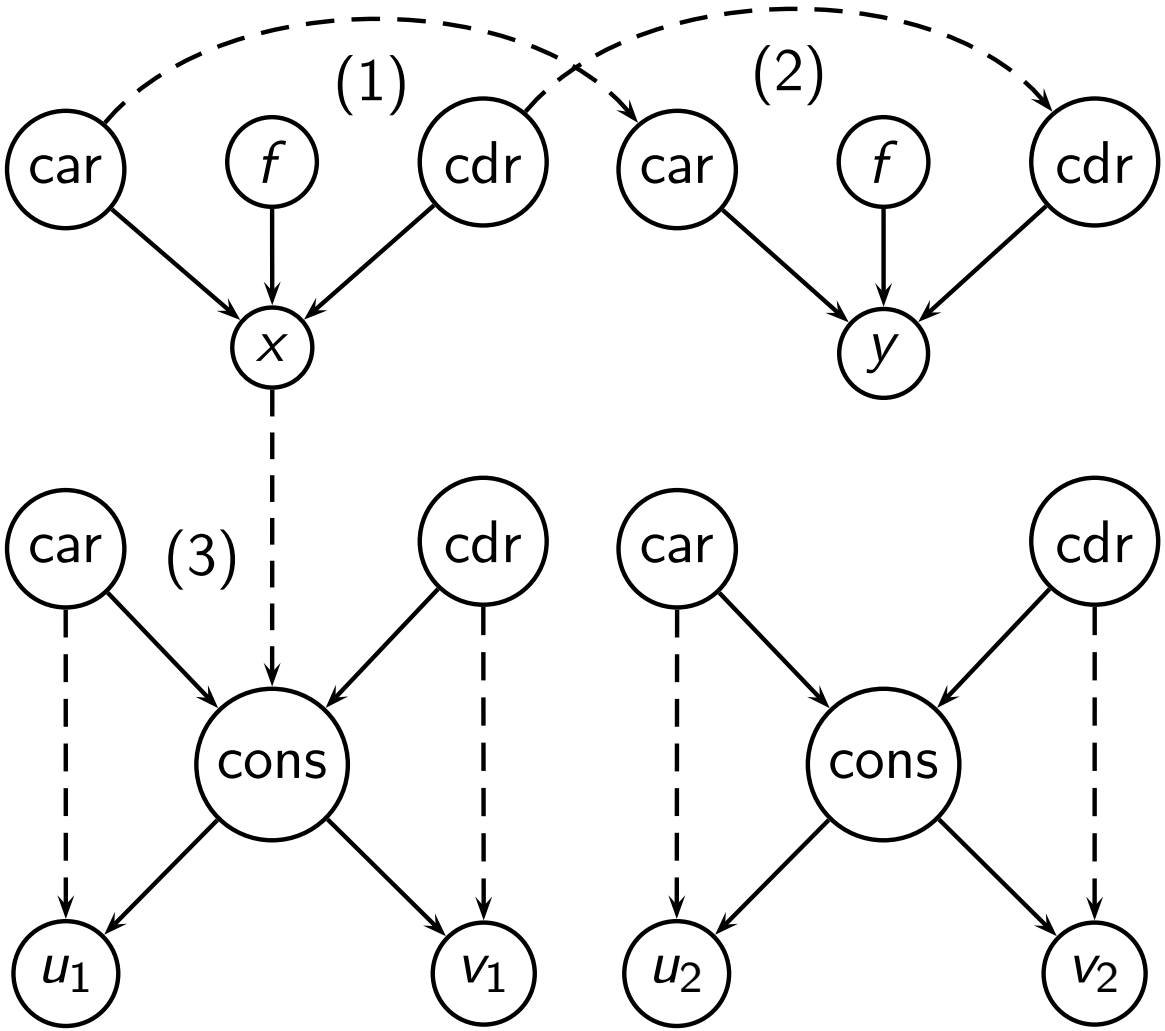
$$\begin{array}{ll} (A1) & \forall x, y.\ \text{car}(\text{cons}(x, y)) = x \\ (A2) & \forall x, y.\ \text{cdr}(\text{cons}(x, y)) = y \end{array}$$

Example(cont): Initial DAG



axioms (A1), (A2)

Example(cont): MERGE



– – explicit equation

. . . by congruence

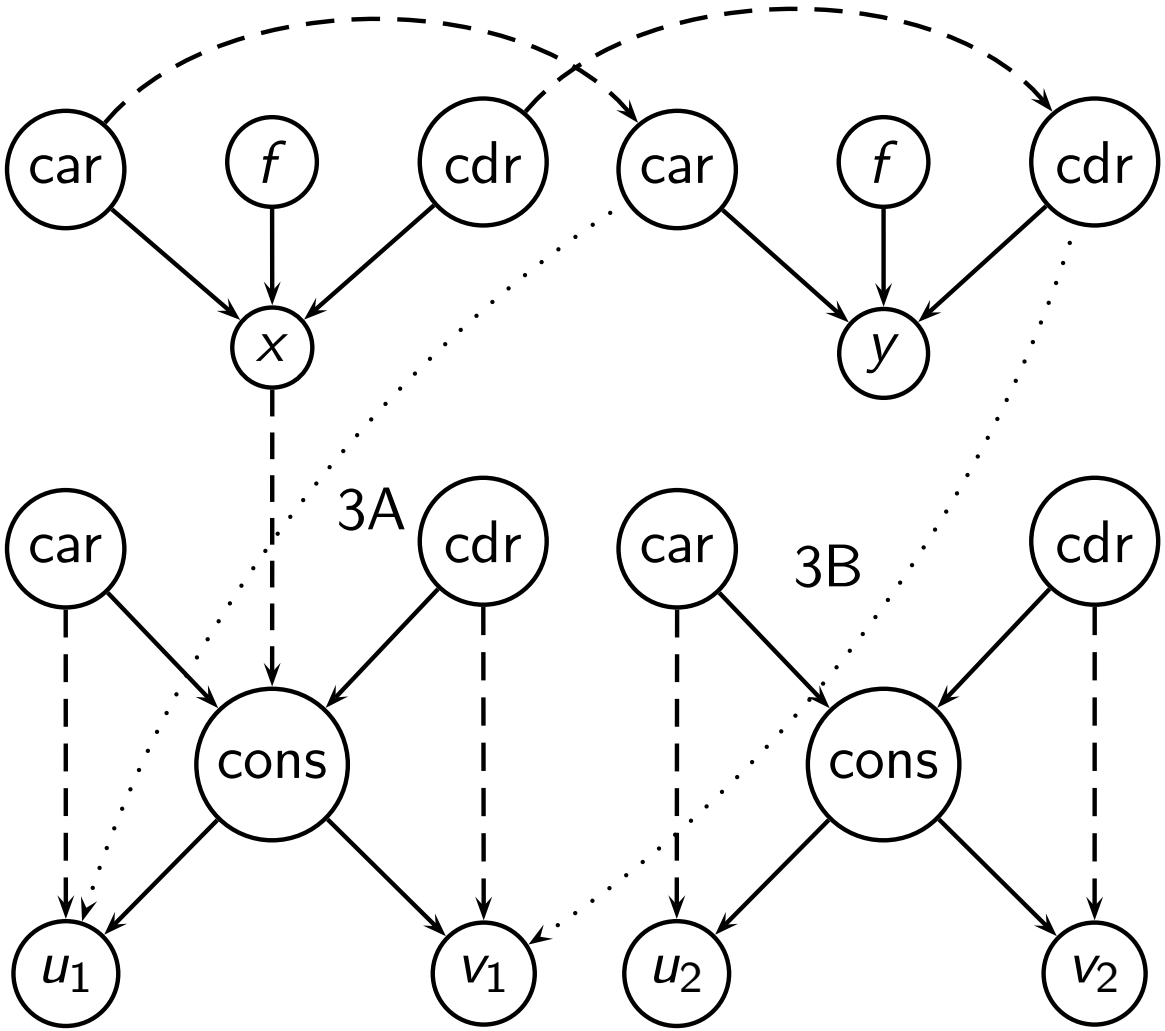
1 : MERGE car(x) car(y)

2 : MERGE cdr(x) cdr(y)

3 : MERGE x cons(u_1, v_1)

$\Downarrow$

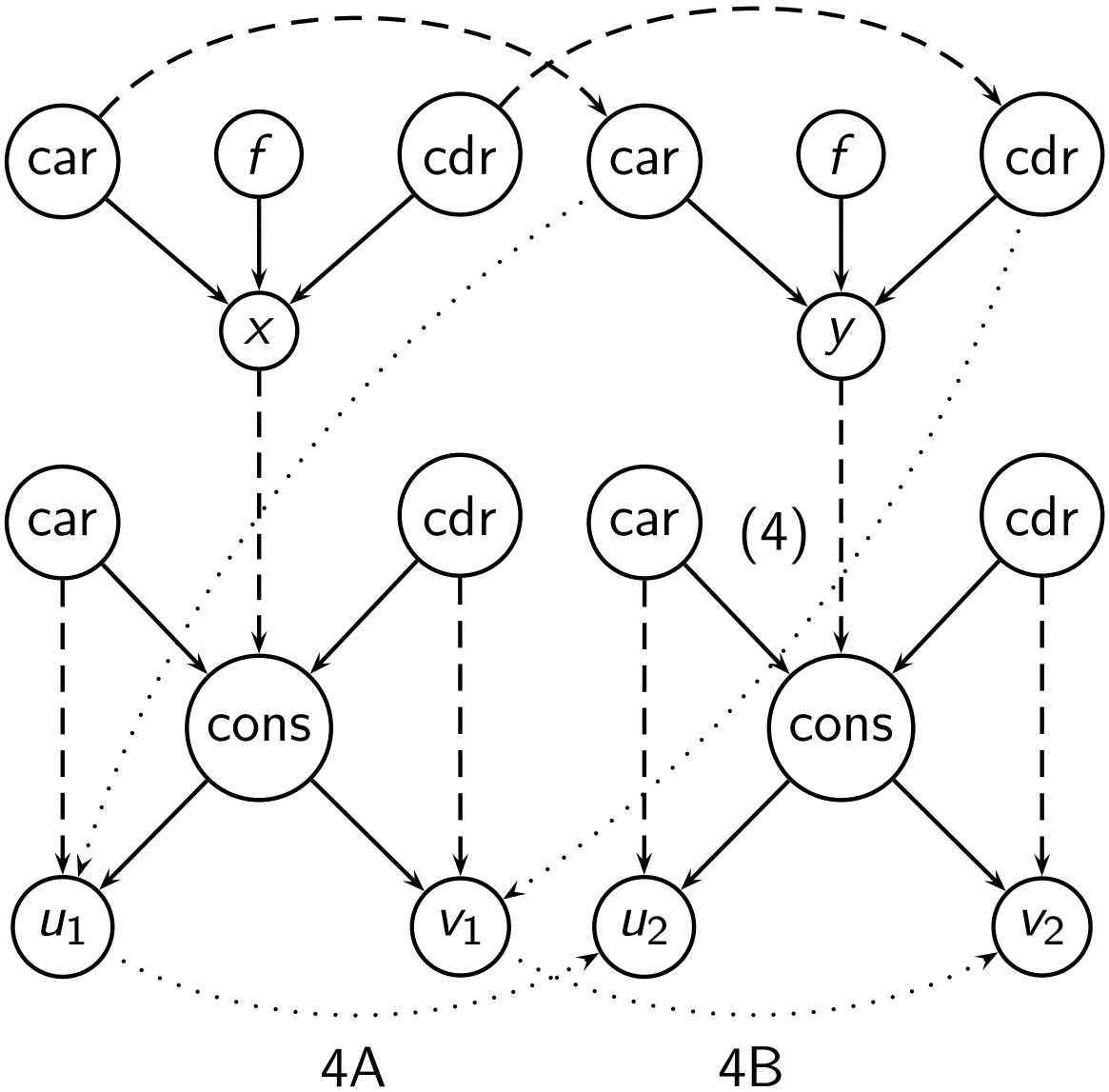
Example(cont): Propagation



Congruent:
car(x) car(cons(u_1, v_1))
FIND car(x) = car(y)
FIND car(cons$(\ldots)$) = u_1

Congruent:
cdr(x) cdr(cons(u_1, v_1))
FIND cdr(x) = cdr(y)
FIND cdr(cons$(\ldots)$) = v_1

Example(cont): MERGE



4 : MERGE y cons(u_2, v_2)

$\Downarrow$

Congruent:

car(y) car(cons(u_2, v_2))

FIND car$(y) = u_1$

FIND car(cons$(\ldots)) = u_2$

Congruent:

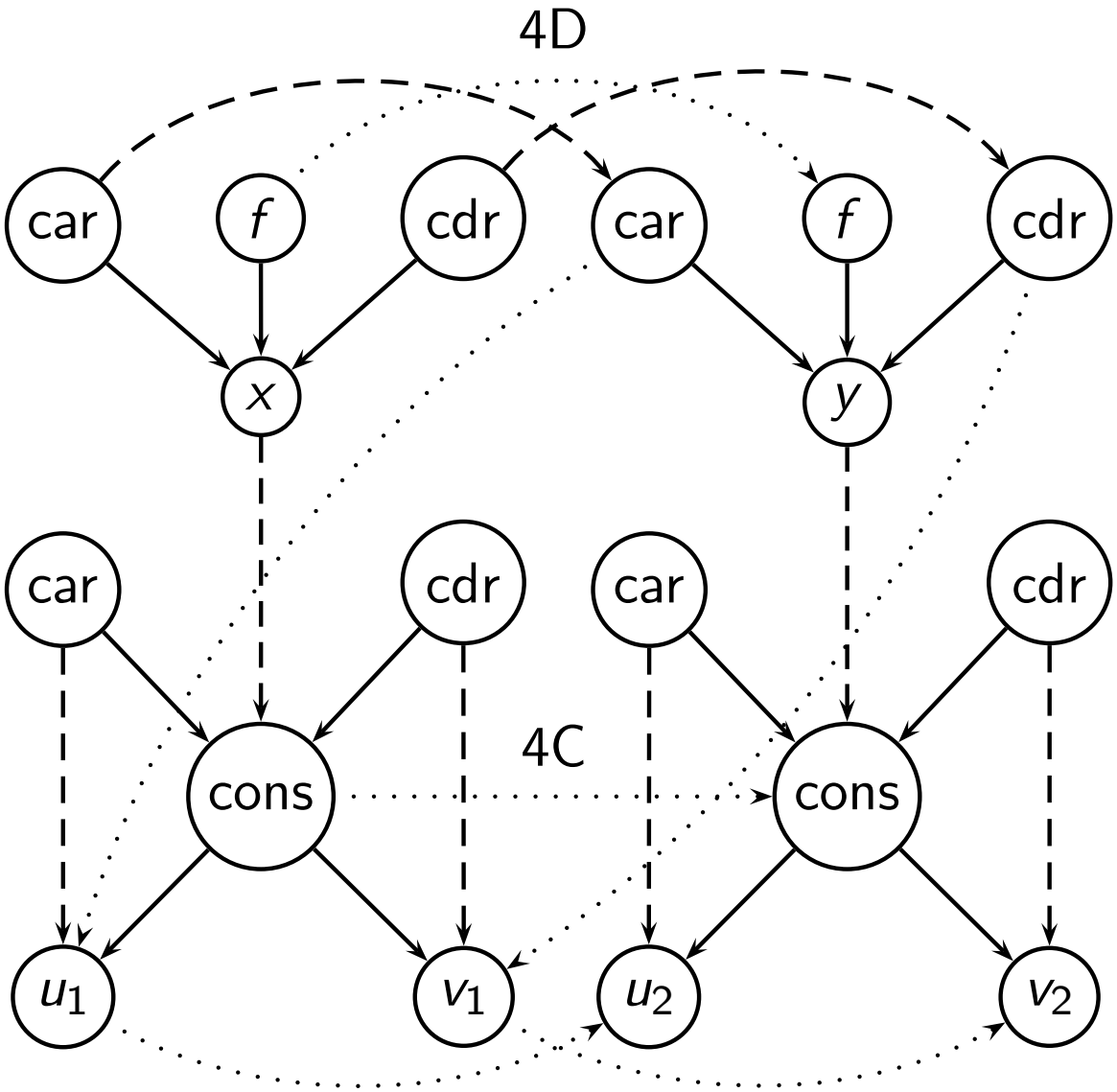
cdr(y) cdr(cons(u_2, v_2))

FIND cdr$(y) = v_1$

FIND cdr(cons$(\ldots)) = v_2$

$\Downarrow$

Example(cont): CONGRUENCE



Congruent:
$\text{cons}(u_1, v_1)$ $\text{cons}(u_2, v_2)$

Congruent: $f(x)$ $f(y)$

5 : $\begin{array}{l} \text{FIND } f(x) = f(y) \\ \text{FIND } f(y) = f(y) \end{array}$

$\Downarrow$

F is **unsatisfiable**

Arrays

(1) Quantifier-free Fragment of T_{A}

$$\Sigma_{\mathsf{A}} : \ \{\cdot[\cdot],\ \cdot\langle\cdot \lhd \cdot\rangle,\ =\}\ ,$$

where

- $a[i]$ is a binary function representing read of array a at index i;
- $a\langle i \lhd v\rangle$ is a ternary function representing write of value v to index i of array a;
- $=$ is a binary predicate.

Axioms of T_{A}:

1. axioms of (reflexivity), (symmetry), and (transitivity) of T_{E}
2. $\forall a, i, j.\ i = j \ \rightarrow\ a[i] = a[j]$ (array congruence)
3. $\forall a, v, i, j.\ i = j \ \rightarrow\ a\langle i \lhd v\rangle[j] = v$ (read-over-write 1)
4. $\forall a, v, i, j.\ i \neq j \ \rightarrow\ a\langle i \lhd v\rangle[j] = a[j]$ (read-over-write 2)

Note: a may itself be a write term, e.g., $a\langle i' \lhd v'\rangle$. Then

$$(a\langle i' \lhd v'\rangle)\langle i \lhd v\rangle$$

means: first write the value v' to index i' of a

then write the value v to index i of a

The Decision Procedure

Given quantifier-free conjunctive Σ_A-formula F.
To decide the T_A-satisfiability of F:

Step 1

If F does not contain any write terms $a\langle i \triangleleft v \rangle$, then

1. associate array variables a with fresh function symbol f_a, and replace read terms $a[i]$ with $f_a(i)$;
2. decide the T_E-satisfiability of the resulting formula.

Step 2

Select some read-over-write term $a\langle i \lhd v\rangle[j]$ (note that a may itself be a write term) and split on two cases:

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

$$F[a\langle i \lhd v\rangle[j]] \quad \text{with} \quad F_1 : \; F[v] \; \wedge \; i = j \; ,$$

   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

$$F[a\langle i \lhd v\rangle[j]] \quad \text{with} \quad F_2 : \; F[a[j]] \; \wedge \; i \neq j \; ,$$

   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: Consider Σ_A-formula

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

F contains a write term,

$$a\langle i_1 \lhd v_1 \rangle \langle i_2 \lhd v_2 \rangle [j] \neq a[j] \; .$$

According to (read-over-write 1), assume $i_2 = j$ and recurse on

$$F_1: \; i_2 = j \; \wedge \; i_1 = j \; \wedge \; i_1 \neq i_2 \; \wedge \; a[j] = v_1 \; \wedge \; v_2 \neq a[j] \; .$$

F_1 does not contain any write terms, so rewrite it to

$$F_1': \; i_2 = j \; \wedge \; i_1 = j \; \wedge \; i_1 \neq i_2 \; \wedge \; f_a(j) = v_1 \; \wedge \; v_2 \neq f_a(j) \; .$$

The first two literals imply that $i_1 = i_2$, contradicting the third literal, so F_1' is T_E-unsatisfiable.

Returning, we try the second case:
according to (read-over-write 2), assume $\underline{i_2 \neq j}$ and recurse on

$$F_2 : \; i_2 \neq j \; \wedge \; i_1 = j \; \wedge \; i_1 \neq i_2 \; \wedge \; a[j] = v_1 \; \wedge \; a\langle i_1 \triangleleft v_1 \rangle[j] \neq a[j] \; .$$

F_2 contains a write term. According to (read-over-write 1), assume $\underline{i_1 = j}$ and recurse on

$$F_3 : \; i_1 = j \; \wedge \; i_2 \neq j \; \wedge \; i_1 = j \; \wedge \; i_1 \neq i_2 \; \wedge \; a[j] = v_1 \; \wedge \; v_1 \neq a[j] \; .$$

Contradiction because of the final two terms. Thus, according to (read-over-write 2), assume $\underline{i_1 \neq j}$ and recurse on

$$F_4 : \; i_1 \neq j \; \wedge \; i_2 \neq j \; \wedge \; i_1 = j \; \wedge \; i_1 \neq i_2 \; \wedge \; a[j] = v_1 \; \wedge \; a[j] \neq a[j] \; .$$

Two contradictions: the first and third literals contradict each other, and the final literal is contradictory. As all branches have been tried, F is T_{A}-unsatisfiable.

Suppose instead that F does not contain the literal $i_1 \neq i_2$. Is this new formula T_{A}-satisfiable?