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
- \(f(x) = f(y) \wedge x \neq y\) \(T_E\)-satisfiable
- \(f(f(f(a))) = a \wedge f(f(f(f(f(a))))) = a \wedge f(a) \neq a\) \(T_E\)-unsatisfiable
- \(x = g(y, z) \rightarrow f(x) = f(g(y, z))\) \(T_E\)-valid
Axioms of $T_E$

1. $\forall x. x = x$  
   (reflexivity)

2. $\forall x, y. x = y \rightarrow y = x$  
   (symmetry)

3. $\forall x, y, z. x = y \land y = z \rightarrow x = z$  
   (transitivity)

define $=$ to be an equivalence relation.

Axiom schema

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

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

For example, for unary $f$, the axiom is

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

Therefore,

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

is $T_E$-valid. ($x' \rightarrow x, y' \rightarrow g(y, z)$).
Axiom schema

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

\[
\forall \bar{x}, \bar{y}. \left( \bigwedge_{i=1}^{n} x_i = y_i \right) \rightarrow (p(\bar{x}) \leftrightarrow p(\bar{y}))
\]  

Thus, for unary \( p \), the axiom is

\[
\forall x', y'. x' = y' \rightarrow (p(x') \leftrightarrow p(y'))
\]

Therefore,

\[
a = b \rightarrow (p(a) \leftrightarrow p(b))
\]

is \( T_E \)-valid. \((x' \rightarrow a, y' \rightarrow b)\).
We discuss $T_{E}$-formulae without predicates

For example, for $\Sigma_{E}$-formula

\[ F : \ p(x) \land q(x, y) \land q(y, z) \rightarrow \neg q(x, z) \]

introduce fresh constant $\bullet$ and fresh functions $f_{p}$ and $f_{q}$, and transform $F$ to

\[ G : \ f_{p}(x) = \bullet \land f_{q}(x, y) = \bullet \land f_{q}(y, z) = \bullet \rightarrow f_{q}(x, z) \neq \bullet . \]
Equivalence and Congruence Relations: Basics

Binary relation $R$ over set $S$

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

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

$m \equiv_2 n \iff (m \mod 2) = (n \mod 2)$

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 \{\text{equivalence, congruence}\} relation $R$ over set $S$, the \{\text{equivalence, congruence}\} class of $s \in S$ under $R$ is

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

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 partition $P$ of $S$ is a set of subsets of $S$ that is

- total $\left( \bigcup_{S' \in P} S' \right) = S$
- disjoint $\forall S_1, S_2 \in P. S_1 \neq S_2 \rightarrow S_1 \cap S_2 = \emptyset$
The quotient $S/R$ of $S$ by \{ equivalence congruence \} relation $R$ is the partition of $S$ into \{ equivalence congruence \} classes

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

It satisfies total and disjoint conditions.

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 a refinement of $R_2$, $R_1 ≺ R_2$, 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 $R_1 ≺ R_2$

- For set $\mathbb{Z}$
  $$R_1 : \{xR_1 y : x \mod 2 = y \mod 2\}$$
  $$R_2 : \{xR_2 y : x \mod 4 = y \mod 4\}$$
  Then $R_2 ≺ R_1$.
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. $R \preceq R^E$;
- for all other equivalence relations $R'$ s.t. $R \preceq R'$, either $R' = R^E$ or $R^E \preceq R'$

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

- $aR^Eb, bR^Ec, dR^Ed$ since $R \subseteq R^E$;
- $aR^Ea, bR^Eb, cR^Ec$ by reflexivity;
- $bR^Ea, cR^Eb$ by symmetry;
- $aR^Ec$ by transitivity;
- $cR^Ea$ by symmetry.

Similarly, the congruence closure $R^C$ of $R$ is the “smallest” congruence relation that “covers” $R$. 
**$T_E$-satisfiability and Congruence Classes I**

**Definition:** For $\Sigma_E$-formula

\[ F : s_1 = t_1 \land \cdots \land s_m = t_m \land s_{m+1} \neq t_{m+1} \land \cdots \land s_n \neq t_n \]

the **subterm set** $S_F$ of $F$ is the set that contains precisely the subterms of $F$.

**Example:** The subterm set of

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

is

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

**Note:** we consider only quantifier-free conjunctive $\Sigma_E$-formulae. Convert non-conjunctive formula $F$ to DNF $\bigvee_i F_i$, where each disjunct $F_i$ is a conjunction of $=, \neq$. Check each disjunct $F_i$. $F$ is $T_E$-satisfiable iff at least one disjunct $F_i$ is $T_E$-satisfiable.
Given $\Sigma_E$-formula $F$

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

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 over $S_F$, or to prove that no congruence relation exists.
Congruence Closure Algorithm

\[ F: \quad s_1 = t_1 \land \cdots \land s_m = t_m \land s_{m+1} \neq t_{m+1} \land \cdots \land s_n \neq t_n \]

generate congruence closure \( \sim \)

search for contradiction

Decide if \( F \) is \( T_E \)-satisfiable.

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

\[ \sim \models s_1 = t_1 \land \cdots \land s_m = t_m. \]

2. If for any \( i \in \{m + 1, \ldots, n\}, s_i \sim t_i \), return unsatisfiable.

3. Otherwise, \( \sim \models F \), so return satisfiable.

How do we actually construct the congruence closure in Step 1?
Congruence Closure Algorithm (Details)

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

\[
\{ \{ s \} : s \in S_F \}.
\]

That is, let each term over \( S_F \) 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 \( s_i \sim t_i \).
Congruence Closure Algorithm: Example 1

Given $\Sigma_E$-formula

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

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 (function) congruence axiom,

$$f(a, b) \sim a, b \sim b \text{ implies } f(f(a, b), b) \sim f(a, b),$$

resulting in the new partition

3. $\{\{a, f(a, b), f(f(a, b), b)\}, \{b\}\}$
This partition represents the congruence closure of $S_F$. Is it the case that

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

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 $\Sigma_E$-formula

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

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

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

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

deduce the following congruence propagations:

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

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. $\{\{a, f^3(a)\}, \{f(a), f^4(a)\}, \{f^2(a), f^5(a)\}\}$.
Congruence Closure Algorithm: Example 2 II

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,

\{\{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.
Given $\Sigma_E$-formula

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

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

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

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

Directed Acyclic Graph (DAG)

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

$\begin{align*}
1 : f & \quad f(f(a, b), b) \\
2 : f & \quad f(a, b) \\
3 : a & \quad a \\
4 : b & \quad b \\
\end{align*}$

Efficient way for computing the congruence closure.
Summary of idea

\[ f(a, b) = a \land f(f(a, b), b) \neq a \]

Initial DAG

\[ f(a, b) = a \Rightarrow \text{MERGE } f(a, b) a \]

\[ f(a, b) \sim a, b \sim b \Rightarrow \text{MERGE } f(f(a, b), b) \sim f(a, b) \]

\[ \text{--- explicit equation} \quad \ldots \ldots \text{by congruence} \]

\[
\begin{align*}
\text{FIND } f(f(a, b), b) &= a = \text{FIND } a \\
f(f(a, b), b) &\neq a
\end{align*}
\]

\[ \Rightarrow \text{Unsatisfiable} \]
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

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

![DAG Diagram]

1 : f

2 : f

3 : a

4 : b
DAG Representation of node 3

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

1 : f

2 : f

3 : a

4 : b
The Implementation I

**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 \( \{2, 3\} \).
**The Implementation II**

**UNION function**

\[
\text{let } \text{UNION } i_1 \ i_2 = \\
\text{let } n_1 = \text{NODE (FIND } i_1) \text{ in} \\
\text{let } n_2 = \text{NODE (FIND } i_2) \text{ in} \\
n_1.\text{find} \leftarrow n_2.\text{find}; \\
n_2.\text{ccpar} \leftarrow n_1.\text{ccpar} \cup n_2.\text{ccpar}; \\
n_1.\text{ccpar} \leftarrow \emptyset \\
\]

\(n_2\) is the representative of the union class
The Implementation III

Example

```
UNION 1 2  \quad n_1 = 1 \quad n_2 = 3
1.find \leftarrow 3
3.ccpar \leftarrow \{1, 2\}
1.ccpar \leftarrow \emptyset
```
The Implementation IV

CCPAR function
Returns parents of all nodes in i’s congruence class

let CCPAR i =
(NODE (FIND i)).ccpar

CONGRUENT predicate
Test whether i₁ and i₂ are congruent

let CONGRUENT i₁ i₂ =
let n₁ = NODE i₁ in
let n₂ = NODE i₂ in
n₁.fn = n₂.fn
∧ |n₁.args| = |n₂.args|
∧ ∀i ∈ {1, . . . , |n₁.args|}. FIND n₁.args[i] = FIND n₂.args[i]
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.
The Implementation VI

**MERGE function**

\[
\begin{align*}
\text{let rec MERGE } & i_1 i_2 = \\
& \quad \text{if FIND } i_1 \neq \text{FIND } i_2 \text{ then begin} \\
& \quad \quad \text{let } P_{i_1} = \text{CCPAR } i_1 \text{ in} \\
& \quad \quad \text{let } P_{i_2} = \text{CCPAR } i_2 \text{ in} \\
& \quad \quad \text{UNION } i_1 i_2; \\
& \quad \quad \text{foreach } t_1 \in P_{i_1}, t_2 \in P_{i_2} \text{ do} \\
& \quad \quad \quad \text{if FIND } t_1 \neq \text{FIND } t_2 \land \text{CONGRUENT } t_1 t_2 \\
& \quad \quad \quad \text{then MERGE } t_1 t_2 \\
& \quad \quad \text{done} \\
& \quad \text{end}
\end{align*}
\]

\( P_{i_1} \) and \( P_{i_2} \) store the values of \( \text{CCPAR } i_1 \) and \( \text{CCPAR } i_2 \) (before the union).
Decision Procedure: $T_E$-satisfiability

Given $\Sigma_E$-formula

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

with subterm set $S_F$, perform the following steps:

1. Construct the initial DAG for the subterm set $S_F$.
2. For $i \in \{1, \ldots, m\}$, MERGE $s_i$ $t_i$.
3. If FIND $s_i = $ 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

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

(1) \hspace{2cm} (2) \hspace{2cm} (3)

Initial DAG

MERGE 2 3

P_2 = \{1\}
P_3 = \{2\}

UNION 2 3

CONGRUENT 1 2

FIND $f(f(a, b), b) = a \Rightarrow$ Unsatisfiable
Given \( \Sigma_E \)-formula

\[
F : f(a, b) = a \land f(f(a, b), b) \neq a .
\]

The subterm set is

\[
S_F = \{ a, b, f(a, b), f(f(a, b), b) \} ,
\]

resulting in the initial partition

\[
(1) \quad \{ \{ a \}, \{ b \}, \{ f(a, b) \}, \{ f(f(a, b), b) \} \}
\]

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

Final partition (Fig (3))

\[
(2) \quad \{ \{ a, f(a, b), f(f(a, b), b) \}, \{ b \} \}
\]

**Note:** dash edge \_\_\_\_ merge dictated by equalities in \( F \)
dotted edge \_\_\_\_\_ deduced merge

Does

\[
\{ \{ a, f(a, b), f(f(a, b), b) \}, \{ b \} \} \models F ?
\]

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

\[
f(f(f(a))) = a \land f(f(f(f(a)))) = a \land f(a) \neq a
\]

\[
\begin{array}{cccccc}
5 : f & \rightarrow & 4 : f & \rightarrow & 3 : f & \rightarrow & 2 : f & \rightarrow & 1 : f & \rightarrow & 0 : a \\
\end{array}
\]

(1)

Initial DAG

\[
f(f(f(a))) = a \Rightarrow \text{MERGE 3 0: } P_3 = \{4\} \quad P_0 = \{1\} \quad \text{UNION 3 0}
\]

\[
\Rightarrow \text{MERGE 4 1: } P_4 = \{5\} \quad P_1 = \{2\} \quad \text{UNION 4 1}
\]

\[
\Rightarrow \text{MERGE 5 2: } P_5 = \{\} \quad P_2 = \{3\} \quad \text{UNION 5 2}
\]
Example 2: $T_E$-Satisfiability

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

5 : \textcolor{red}{f} \rightarrow 4 : \textcolor{red}{f} \rightarrow 3 : \textcolor{red}{f} \rightarrow 2 : \textcolor{red}{f} \rightarrow 1 : \textcolor{red}{f} \rightarrow 0 : a \quad (2)$$

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

$$\text{UNION} \ 5 \ 0$$

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

$$\text{UNION} \ 3 \ 1$$

FIND $f(a) = f(a) = \text{FIND} \ a \Rightarrow \textbf{Unsatisfiable}$
Given $\Sigma_E$-formula

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

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 $\Sigma_E$-formula $F$ is $T_E$-satisfiable iff the congruence closure algorithm returns satisfiable.
Recursive Data Structures

Quantifier-free Theory of Lists $T_{\text{cons}}$

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

- **constructor** $\text{cons} : \text{cons}(x, y)$ list constructed by appending $y$ to $x$
- **left projector** $\text{car} : \text{car}(\text{cons}(x, y)) = x$
- **right projector** $\text{cdr} : \text{cdr}(\text{cons}(x, y)) = y$
- **atom** : unary predicate
Axioms of $T_{\text{cons}}$

- reflexivity, symmetry, transitivity
- function (congruence) axioms:

  \[
  \forall x_1, x_2, y_1, y_2. \ x_1 = x_2 \land y_1 = y_2 \rightarrow \text{cons}(x_1, y_1) = \text{cons}(x_2, y_2)
  \]

  \[
  \forall x, y. \ x = y \rightarrow \text{car}(x) = \text{car}(y)
  \]

  \[
  \forall x, y. \ x = y \rightarrow \text{cdr}(x) = \text{cdr}(y)
  \]

- predicate (congruence) axiom:

  \[
  \forall x, y. \ x = y \rightarrow (\text{atom}(x) \iff \text{atom}(y))
  \]

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

- Consider only quantifier-free conjunctive $\Sigma_{cons}$-formulae. Convert non-conjunctive formula to DNF and check each disjunct.

- $\neg \text{atom}(u_i)$ literals are removed:

\[
\text{replace } \neg \text{atom}(u_i) \text{ with } u_i = \text{cons}(u_{i1}, u_{i2})
\]

by the (construction) axiom.

- Result of a conjunctive $\Sigma_{cons}$-formula with literals

\[
s = t \quad s \neq t \quad \text{atom}(u)
\]

- Because of similarity to $\Sigma_E$, we sometimes combine $\Sigma_{cons} \cup \Sigma_E$. 

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

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

where $s_i$, $t_i$, and $u_i$ are $T_{\text{cons}}$-terms
Algorithm: $T_{\text{cons}}$-Satisfiability

1. Construct the initial DAG for $S_F$
2. for each node $n$ with $n.fn = \text{cons}$
   - add $\text{car}(n)$ and $\text{merge} \text{car}(n) n.\text{args}[1]$
   - add $\text{cdr}(n)$ and $\text{merge} \text{cdr}(n) n.\text{args}[2]$
   by axioms (A1), (A2)
3. for $1 \leq i \leq m$, $\text{merge} \ s_i \ t_i$
4. for $m + 1 \leq i \leq n$, if $\text{find} \ s_i = \text{find} \ t_i$, return unsatisfiable
5. for $1 \leq i \leq \ell$, if $\exists v. \text{find} \ v = \text{find} \ u_i \land v.fn = \text{cons}$, return unsatisfiable
6. Otherwise, return satisfiable
Example

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

\[
F : \quad \text{car}(x) = \text{car}(y) \land \text{cdr}(x) = \text{cdr}(y) \land \\
\quad \neg \text{atom}(x) \land \neg \text{atom}(y) \land f(x) \neq f(y)
\]

where the function symbol \(f\) is in \(\Sigma_{E}\)

\[
\begin{align*}
\text{car}(x) &= \text{car}(y) \quad \land \quad (1) \\
\text{cdr}(x) &= \text{cdr}(y) \quad \land \quad (2) \\
x &= \text{cons}(u_1, v_1) \quad \land \quad (3) \\
y &= \text{cons}(u_2, v_2) \quad \land \quad (4) \\
f(x) &\neq f(y) \quad \land \quad (5)
\end{align*}
\]

Recall the projection axioms:

\[
\begin{align*}
(A1) \quad \forall x, y. \text{car(cons}(x, y)) &= x \\
(A2) \quad \forall x, y. \text{cdr}(\text{cons}(x, y)) &= y
\end{align*}
\]
Example (cont): Initial DAG

\[
\begin{align*}
\text{car} & \quad f & \quad \text{cdr} \\
\quad & \quad x & \\
\text{car} & \quad f & \quad \text{cdr} \\
\quad & \quad y \\
\text{car} & \quad \text{cdr} \\
\quad & \quad \text{cons} \\
\quad & \quad u_1 \\
\text{car} & \quad \text{cdr} \\
\quad & \quad \text{cons} \\
\quad & \quad v_1 \\
\text{car} & \quad \text{cdr} \\
\quad & \quad u_2 \\
\text{car} & \quad \text{cdr} \\
\quad & \quad v_2
\end{align*}
\]

axioms (A1), (A2)
Example (cont): **MERGE**

\[
\begin{align*}
(1) & : \text{MERGE } \text{car}(x) \text{ car}(y) \\
(2) & : \text{MERGE } \text{cdr}(x) \text{ cdr}(y) \\
(3) & : \text{MERGE } x \text{ cons}(u_1, v_1)
\end{align*}
\]

\[\downarrow\]

\[\text{-- explicit equation} \]

\[\ldots \text{ by congruence} \]

\[1 : \text{MERGE } \text{car}(x) \text{ car}(y) \]

\[2 : \text{MERGE } \text{cdr}(x) \text{ cdr}(y) \]

\[3 : \text{MERGE } x \text{ cons}(u_1, v_1)\]
Example (cont): Propagation

Congruent:
\[ \text{car}(x) \equiv \text{car}(\text{cons}(u_1, v_1)) \]
\[ \text{FIND} \quad \text{car}(x) = \text{car}(y) \]
\[ \text{FIND} \quad \text{car}(\text{cons}(\ldots)) = u_1 \]

Congruent:
\[ \text{cdr}(x) \equiv \text{cdr}(\text{cons}(u_1, v_1)) \]
\[ \text{FIND} \quad \text{cdr}(x) = \text{cdr}(y) \]
\[ \text{FIND} \quad \text{cdr}(\text{cons}(\ldots)) = v_1 \]
Example (cont): **MERGE**

\[
\begin{align*}
4 : & \text{MERGE } y \text{ cons}(u_2, v_2) \\
& \Downarrow \\
\text{Congruent: } \\
\text{car}(y) & \text{ car}(\text{cons}(u_2, v_2)) \\
\text{FIND car}(y) & = u_1 \\
\text{FIND car}(\text{cons}(\ldots)) & = u_2 \\
\text{Congruent: } \\
\text{cdr}(y) & \text{ cdr}(\text{cons}(u_2, v_2)) \\
\text{FIND cdr}(y) & = v_1 \\
\text{FIND cdr}(\text{cons}(\ldots)) & = v_2 \\
& \Downarrow
\end{align*}
\]
Example (cont): CONGRUENCE

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

Congruent:
\[ f(x) \quad f(y) \]

5: \begin{align*}
  \text{FIND} & \quad f(x) = f(y) \\
  \text{FIND} & \quad f(y) = f(y) \\
\end{align*}

\[ \downarrow \]

\( F \) is unsatisfiable