# Automated Deduction

Laura Kovács
TU Wien

# SMT Related Problems – Where are we now?

- ✓ Deciding theory: Check satisfiability of a set of *literals* in $\mathcal{T}_E$

  What about in $\mathcal{T}_A$ and $\mathcal{T}_Q$?

- ? Put together theory reasoning and SAT solving

- ? What about combination of theories: Given decision procedures for theories, how can we build a decision procedure for formulas using several theories?

# Outline

# Deciding $\mathcal{T}_E$: Problems Using Non-Unit Clauses

Example: Let's try to prove the validity of the formula:

$$F: \ (a = b \vee a = c) \wedge f(a) = a \rightarrow f(f(b)) = a \vee f(c) = a.$$

This is equivalent to establishing unsatisfiability of

$$a = b \vee a = c$$
$$f(a) = a$$
$$f(f(b)) \neq a$$
$$f(c) \neq a$$

# Deciding $\mathcal{T}_E$: Problems Using Non-Unit Clauses

Example: Let's try to prove the validity of the formula:

$$F: \ (a = b \vee a = c) \wedge f(a) = a \rightarrow f(f(b)) = a \vee f(c) = a.$$

This is equivalent to establishing unsatisfiability of

$$a = b \vee a = c$$
$$f(a) = a$$
$$f(f(b)) \neq a$$
$$f(c) \neq a$$

We have a non-unit clause, so we cannot use congruence closure.

Inputs of the congruence closure algorithm are conjunctions of $\mathcal{T}_E$-literals.

# Idea: Use a SAT Solver

Add a propositional symbol to name every theory atom:

$$a = b \vee a = c$$
$$f(a) = a$$
$$f(f(b)) \neq a$$
$$f(c) \neq a$$

$$p_1 \vee p_2$$
$$p_3$$
$$\neg p_4$$
$$\neg p_5$$

$$p_1 : a = b$$
$$p_2 : a = c$$
$$p_3 : f(a) = a$$
$$p_4 : f(f(b)) = a$$
$$p_5 : f(c) = a$$

# Idea: Use a SAT Solver

Add a propositional symbol to name every theory atom:

$$a = b \vee a = c$$
$$f(a) = a$$
$$f(f(b)) \neq a$$
$$f(c) \neq a$$

$$p_1 \vee p_2$$
$$p_3$$
$$\neg p_4$$
$$\neg p_5$$

$$p_1 : a = b$$
$$p_2 : a = c$$
$$p_3 : f(a) = a$$
$$p_4 : f(f(b)) = a$$
$$p_5 : f(c) = a$$

1. Use a SAT solver (DPLL) over the propositional clauses.
2. If the SAT solver returns `unsatisfiable`, $\neg F$ is `unsatisfiable`.
3. If the SAT solver returns `satisfiable`, we obtain a set of literals $L_1, \ldots, L_n$ representing a model $I$ of the propositional clauses.

# Idea: Use a SAT Solver

Add a propositional symbol to name every theory atom:

$$a = b \vee a = c$$
$$f(a) = a$$
$$f(f(b)) \neq a$$
$$f(c) \neq a$$

$$p_1 \vee p_2$$
$$p_3$$
$$\neg p_4$$
$$\neg p_5$$

$$p_1 : a = b$$
$$p_2 : a = c$$
$$p_3 : f(a) = a$$
$$p_4 : f(f(b)) = a$$
$$p_5 : f(c) = a$$

1. Use a SAT solver (DPLL) over the propositional clauses.
2. If the SAT solver returns `unsatisfiable`, $\neg F$ is `unsatisfiable`.
3. If the SAT solver returns `satisfiable`, we obtain a set of literals $L_1, \ldots, L_n$ representing a model $I$ of the propositional clauses.
4. Using the theory solver (congruence closure), check satisfiability of the theory literals corresponding to $I$.
5. If the theory solvers returns `satisfiable`, $\neg F$ is `satisfiable`.
6. If the theory solvers returns `unsatisfiable`,

# Idea: Use a SAT Solver

Add a propositional symbol to name every theory atom:

$$a = b \vee a = c$$
$$f(a) = a$$
$$f(f(b)) \neq a$$
$$f(c) \neq a$$

$$p_1 \vee p_2$$
$$p_3$$
$$\neg p_4$$
$$\neg p_5$$

$$p_1 : a = b$$
$$p_2 : a = c$$
$$p_3 : f(a) = a$$
$$p_4 : f(f(b)) = a$$
$$p_5 : f(c) = a$$

1. Use a SAT solver (DPLL) over the propositional clauses.
2. If the SAT solver returns `unsatisfiable`, $\neg F$ is `unsatisfiable`.
3. If the SAT solver returns `satisfiable`, we obtain a set of literals $L_1, \ldots, L_n$ representing a model $I$ of the propositional clauses.
4. Using the theory solver (congruence closure), check satisfiability of the theory literals corresponding to $I$.
5. If the theory solvers returns `satisfiable`, $\neg F$ is `satisfiable`.
6. If the theory solvers returns `unsatisfiable`, add $\neg L_1 \vee \ldots \vee \neg L_n$ to the set of propositional clauses and go back to step 1.

# Idea: Use a SAT Solver

Add a propositional symbol to name every theory atom:

$$a = b \vee a = c$$
$$f(a) = a$$
$$f(f(b)) \neq a$$
$$f(c) \neq a$$

$$p_1 \vee p_2$$
$$p_3$$
$$\neg p_4$$
$$\neg p_5$$

$$p_1 : a = b$$
$$p_2 : a = c$$
$$p_3 : f(a) = a$$
$$p_4 : f(f(b)) = a$$
$$p_5 : f(c) = a$$

**DPLL($\mathcal{T}$) Algorithm for SMT**

1. Use a SAT solver (DPLL) over the propositional clauses.
2. If the SAT solver returns `unsatisfiable`, $\neg F$ is `unsatisfiable`.
3. If the SAT solver returns `satisfiable`, we obtain a set of literals $L_1, \ldots, L_n$ representing a model $I$ of the propositional clauses.
4. Using the theory solver (congruence closure), check satisfiability of the theory literals corresponding to $I$.
5. If the theory solvers returns `satisfiable`, $\neg F$ is `satisfiable`.
6. If the theory solvers returns `unsatisfiable`, add $\neg L_1 \vee \ldots \vee \neg L_n$ to the set of propositional clauses and go back to step 1.

# Problems – Where are we now?

- ✓ Deciding theory: Check satisfiability of a set of *literals* in $\mathcal{T}_E$
- ✓ Put together theory reasoning and SAT solving

# Problems – Where are we now?

- ✓ Deciding theory: Check satisfiability of a set of *literals* in $\mathcal{T}_E$

  What about in $\mathcal{T}_A$ and $\mathcal{T}_Q$?

- ✓ Put together theory reasoning and SAT solving

# Problems – Where are we now?

- ✓ Deciding theory: Check satisfiability of a set of *literals* in $\mathcal{T}_E$

  What about in $\mathcal{T}_A$ and $\mathcal{T}_Q$?

- ✓ Put together theory reasoning and SAT solving

- ? What about combination of theories: Given decision procedures for theories, how can we build a decision procedure for formulas using several theories?

# Outline

# Deciding $\mathcal{T}_A$

The theory of arrays $\mathcal{T}_A$ is defined by

- a signature $\Sigma_A = \{read, write\}$ where

  $read$ is a binary function: $read(A, x)$ is the value of array $A$ at position $x$

  $write$ is a ternary function: $write(A, x, v)$ is the modified array $A$ in which the element at position $x$ has value $v$

# Deciding $\mathcal{T}_A$

The theory of arrays $\mathcal{T}_A$ is defined by

- a signature $\Sigma_A = \{read, write\}$ where

  $read$ is a binary function: $read(A, x)$ is the value of array $A$ at position $x$

  $write$ is a ternary function: $write(A, x, v)$ is the modified array $A$ in which the element at position $x$ has value $v$

  From now on, we consider $=$ as part of the language.

# Deciding $\mathcal{T}_A$

The theory of arrays $\mathcal{T}_A$ is defined by

- a signature $\Sigma_A = \{read, write\}$ where

  $read$ is a binary function: $read(A, x)$ is the value of array $A$ at position $x$

  $write$ is a ternary function: $write(A, x, v)$ is the modified array $A$ in which the element at position $x$ has value $v$

  From now on, we consider $=$ as part of the language.

- the following axioms:

  *equality axioms* from $\mathcal{T}_E$

  $$x = y \rightarrow read(write(A, x, v), y) = v \qquad \text{(read-over-write 1)}$$

  $$x \neq y \rightarrow read(write(A, x, v), y) = read(A, y) \qquad \text{(read-over-write 2)}$$

# Deciding $\mathcal{T}_A$: Congruence Closure Algorithm

The theory of arrays $\mathcal{T}_A$ is defined by

- a signature $\Sigma_A = \{read, write\}$ where

  $read$ is a binary function: $read(A, x)$ is the value of array $A$ at position $x$

  $write$ is a ternary function: $write(A, x, v)$ is the modified array $A$ in which the element at position $x$ has value $v$

  From now on, we consider $=$ as part of the language.

- the following axioms:

  *equality axioms* from $\mathcal{T}_E$

$$x = y \rightarrow read(write(A, x, v), y) = v \qquad \text{(read-over-write 1)}$$

$$x \neq y \rightarrow read(write(A, x, v), y) = read(A, y) \qquad \text{(read-over-write 2)}$$

$\mathcal{T}_A$-satisfiability of a formula $F$ is reduced to $\mathcal{T}_E$-satisfiability

# Deciding $\mathcal{T}_A$: Congruence Closure Algorithm

The theory of arrays $\mathcal{T}_A$ is defined by

- a signature $\Sigma_A = \{read, write\}$ where

  $read$ is a binary function: $read(A, x)$ is the value of array $A$ at position $x$

  $write$ is a ternary function: $write(A, x, v)$ is the modified array $A$ in which the element at position $x$ has value $v$

  From now on, we consider $=$ as part of the language.

- the following axioms:

  *equality axioms* from $\mathcal{T}_E$

$$x = y \rightarrow read(write(A, x, v), y) = v \qquad \text{(read-over-write 1)}$$

$$x \neq y \rightarrow read(write(A, x, v), y) = read(A, y) \qquad \text{(read-over-write 2)}$$

$\mathcal{T}_A$-satisfiability of a formula $F$ is reduced to $\mathcal{T}_E$-satisfiability

Idea:

1. $F$ contains no $write$-terms. Then, treat $read$-terms as uninterpreted function terms
2. $F$ contains $write$-terms. Then, $write$-terms occur in the context of a $read$-term, so use (read-over-write) axioms to "eliminate" $write$-terms

# Deciding $\mathcal{T}_A$: Congruence Closure Algorithm

## Algorithm for deciding $\mathcal{T}_A$

1. $F$ contains no *write*-terms:
   - use fresh function symbol $f_A$ for array variables $A$
   - replace $read(A, x)$ with $f_A(x)$ in $F$
   - decide and return $\mathcal{T}_E$-satisfiability of resulting formula

# Deciding $\mathcal{T}_A$: Congruence Closure Algorithm

## Algorithm for deciding $\mathcal{T}_A$

1. $F$ contains no *write*-terms:

   - use fresh function symbol $f_A$ for array variables $A$
   - replace $read(A, x)$ with $f_A(x)$ in $F$
   - decide and return $\mathcal{T}_E$-satisfiability of resulting formula

2. $F$ contains *write*-terms, say $read(write(A, x, v), y)$

   - Using (read-over-write 1), replace $F$ by the following formula $F_1$:

     $$F_1 : x = y \wedge F[v]$$

     where $F[v]$ denotes the formula obtained by replacing $read(write(A, x, v), y)$ with $v$ in $F$.

# Deciding $\mathcal{T}_A$: Congruence Closure Algorithm

## Algorithm for deciding $\mathcal{T}_A$

1. $F$ contains no *write*-terms:

   - use fresh function symbol $f_A$ for array variables $A$
   - replace $read(A, x)$ with $f_A(x)$ in $F$
   - decide and return $\mathcal{T}_E$-satisfiability of resulting formula

2. $F$ contains *write*-terms, say $read(write(A, x, v), y)$

   - Using (read-over-write 1), replace $F$ by the following formula $F_1$:

     $$F_1 : x = y \wedge F[v]$$

     where $F[v]$ denotes the formula obtained by replacing $read(write(A, x, v), y)$ with $v$ in $F$. If $F_1$ is $\mathcal{T}_A$-satisfiable, return `satisfiable`

# Deciding $\mathcal{T}_A$: Congruence Closure Algorithm

## Algorithm for deciding $\mathcal{T}_A$

1. $F$ contains no *write*-terms:

   - use fresh function symbol $f_A$ for array variables $A$
   - replace $read(A, x)$ with $f_A(x)$ in $F$
   - decide and return $\mathcal{T}_E$-satisfiability of resulting formula

2. $F$ contains *write*-terms, say $read(write(A, x, v), y)$

   - Using (read-over-write 1), replace $F$ by the following formula $F_1$:

     $$F_1 : x = y \wedge F[v]$$

     where $F[v]$ denotes the formula obtained by replacing $read(write(A, x, v), y)$ with $v$ in $F$. If $F_1$ is $\mathcal{T}_A$-satisfiable, return `satisfiable`

   - Using (read-over-write 2), replace $F$ by the following formula $F_2$:

     $$F_1 : x \neq y \wedge F[read(A, y)]$$

     where $F[read(A, y)]$ denotes the formula by replacing $read(write(A, x, v), y)$ with $read(A, y)$ in $F$.

# Deciding $\mathcal{T}_A$: Congruence Closure Algorithm

## Algorithm for deciding $\mathcal{T}_A$

1. $F$ contains no *write*-terms:

   - use fresh function symbol $f_A$ for array variables $A$
   - replace $read(A, x)$ with $f_A(x)$ in $F$
   - decide and return $\mathcal{T}_E$-satisfiability of resulting formula

2. $F$ contains *write*-terms, say $read(write(A, x, v), y)$

   - Using (read-over-write 1), replace $F$ by the following formula $F_1$:

     $$F_1 : x = y \wedge F[v]$$

     where $F[v]$ denotes the formula obtained by replacing $read(write(A, x, v), y)$ with $v$ in $F$. If $F_1$ is $\mathcal{T}_A$-satisfiable, return `satisfiable`

   - Using (read-over-write 2), replace $F$ by the following formula $F_2$:

     $$F_1 : x \neq y \wedge F[read(A, y)]$$

     where $F[read(A, y)]$ denotes the formula by replacing $read(write(A, x, v), y)$ with $read(A, y)$ in $F$. If $F_2$ is $\mathcal{T}_A$-satisfiable, return `satisfiable`

# Deciding $\mathcal{T}_A$: Congruence Closure Algorithm

## Algorithm for deciding $\mathcal{T}_A$

1. $F$ contains no *write*-terms:

   - use fresh function symbol $f_A$ for array variables $A$
   - replace $read(A, x)$ with $f_A(x)$ in $F$
   - decide and return $\mathcal{T}_E$-satisfiability of resulting formula

2. $F$ contains *write*-terms, say $read(write(A, x, v), y)$

   - Using (read-over-write 1), replace $F$ by the following formula $F_1$:

     $$F_1 : x = y \wedge F[v]$$

     where $F[v]$ denotes the formula obtained by replacing $read(write(A, x, v), y)$ with $v$ in $F$. If $F_1$ is $\mathcal{T}_A$-satisfiable, return `satisfiable`
   - Using (read-over-write 2), replace $F$ by the following formula $F_2$:

     $$F_1 : x \neq y \wedge F[read(A, y)]$$

     where $F[read(A, y)]$ denotes the formula by replacing $read(write(A, x, v), y)$ with $read(A, y)$ in $F$. If $F_2$ is $\mathcal{T}_A$-satisfiable, return `satisfiable`

   If $F_1$ and $F_2$ are $\mathcal{T}_A$-unsatisfiable, return `unsatisfiable`

# Deciding $\mathcal{T}_A$: Congruence Closure by Example

Question: Is formula $F$ given below $\mathcal{T}_A$-satisfiable?

$$x_1 = y \ \wedge \ x_1 \neq x_2 \ \wedge \ read(A, y) = v_1 \ \wedge$$
$$read(write(write(A, x1, v1), x2, v2), y) \neq read(A, y)$$

# Deciding $\mathcal{T}_A$: Congruence Closure by Example

Question: Is formula $F$ given below $\mathcal{T}_A$-satisfiable?

$$x_1 = y \ \wedge \ x_1 \neq x_2 \ \wedge \ read(A, y) = v_1 \ \wedge$$
$$read(write(write(A, x1, v1), x2, v2), y) \neq read(A, y)$$

Use (read-over-write-1), and get:

$$F_1 : x_2 = y \ \wedge \ x_1 = y \ \wedge \ x_1 \neq x_2 \ \wedge \ read(A, y) = v_1 \ \wedge v_2 \neq read(A, y)$$

# Deciding $\mathcal{T}_A$: Congruence Closure by Example

Question: Is formula $F$ given below $\mathcal{T}_A$-satisfiable?

$$x_1 = y \ \wedge \ x_1 \neq x_2 \ \wedge \ read(A, y) = v_1 \ \wedge$$
$$read(write(write(A, x1, v1), x2, v2), y) \neq read(A, y)$$

Use (read-over-write-1), and get:

$$F_1 : x_2 = y \ \wedge \ x_1 = y \ \wedge \ x_1 \neq x_2 \ \wedge \ read(A, y) = v_1 \ \wedge v_2 \neq read(A, y)$$

$F_1$ contains no *write*-terms, so rewrite it to:

$$F_1' : x_2 = y \ \wedge \ x_1 = y \ \wedge \ x_1 \neq x_2 \ \wedge \ f_A(y) = v_1 \ \wedge v_2 \neq f_A(y)$$

# Deciding $\mathcal{T}_A$: Congruence Closure by Example

Question: Is formula $F$ given below $\mathcal{T}_A$-satisfiable?

$$x_1 = y \ \wedge \ x_1 \neq x_2 \ \wedge \ read(A, y) = v_1 \ \wedge$$
$$read(write(write(A, x1, v1), x2, v2), y) \neq read(A, y)$$

Use (read-over-write-1), and get:

$$F_1 : x_2 = y \ \wedge \ x_1 = y \ \wedge \ x_1 \neq x_2 \ \wedge \ read(A, y) = v_1 \ \wedge v_2 \neq read(A, y)$$

$F_1$ contains no *write*-terms, so rewrite it to:

$$F_1' : x_2 = y \ \wedge \ x_1 = y \ \wedge \ x_1 \neq x_2 \ \wedge \ f_A(y) = v_1 \ \wedge v_2 \neq f_A(y)$$

$F_1'$ is $\mathcal{T}_E$-unsatisfiable.

# Deciding $\mathcal{T}_A$: Congruence Closure by Example

Question: Is formula $F$ given below $\mathcal{T}_A$-satisfiable?

$$x_1 = y \ \wedge \ x_1 \neq x_2 \ \wedge \ read(A, y) = v_1 \ \wedge$$
$$read(write(write(A, x1, v1), x2, v2), y) \neq read(A, y)$$

Use (read-over-write-1), and get: `unsatisfiable`

$F_1 : x_2 = y \ \wedge \ x_1 = y \ \wedge \ x_1 \neq x_2 \ \wedge \ read(A, y) = v_1 \ \wedge v_2 \neq read(A, y)$

$F_1$ contains no *write*-terms, so rewrite it to:

$$F_1' : x_2 = y \ \wedge \ x_1 = y \ \wedge \ x_1 \neq x_2 \ \wedge \ f_A(y) = v_1 \ \wedge v_2 \neq f_A(y)$$

$F_1'$ is $\mathcal{T}_E$-unsatisfiable.

# Deciding $\mathcal{T}_A$: Congruence Closure by Example

Question: Is formula $F$ given below $\mathcal{T}_A$-satisfiable?

$$x_1 = y \ \wedge \ x_1 \neq x_2 \ \wedge \ read(A, y) = v_1 \ \wedge$$
$$read(write(write(A, x1, v1), x2, v2), y) \neq read(A, y)$$

Use (read-over-write-1), and get: `unsatisfiable`

Use (read-over-write-2), and get:

$$F_2 : x_2 \neq y \wedge x_1 = y \wedge x_1 \neq x_2 \wedge read(A, y) = v_1 \wedge read(write(A, x_1, v_1), y) \neq read(A, y)$$

# Deciding $\mathcal{T}_A$: Congruence Closure by Example

Question: Is formula $F$ given below $\mathcal{T}_A$-satisfiable?

$$x_1 = y \ \wedge \ x_1 \neq x_2 \ \wedge \ read(A, y) = v_1 \ \wedge$$
$$read(write(write(A, x1, v1), x2, v2), y) \neq read(A, y)$$

Use (read-over-write-1), and get: `unsatisfiable`

Use (read-over-write-2), and get:

$$F_2 : x_2 \neq y \wedge x_1 = y \wedge x_1 \neq x_2 \wedge read(A, y) = v_1 \wedge read(write(A, x_1, v_1), y) \neq read(A, y)$$

Use (read-over-write-1), and get:

# Deciding $\mathcal{T}_A$: Congruence Closure by Example

Question: Is formula $F$ given below $\mathcal{T}_A$-satisfiable?

$$x_1 = y \ \wedge \ x_1 \neq x_2 \ \wedge \ read(A, y) = v_1 \ \wedge$$
$$read(write(write(A, x1, v1), x2, v2), y) \neq read(A, y)$$

Use (read-over-write-1), and get: `unsatisfiable`

Use (read-over-write-2), and get:

$$F_2 : x_2 \neq y \wedge x_1 = y \wedge x_1 \neq x_2 \wedge read(A, y) = v_1 \wedge read(write(A, x_1, v_1), y) \neq read(A, y)$$

Use (read-over-write-1), and get:

$$F_2' : x_1 = y \wedge \ x_2 \neq y \wedge \ x_1 = y \wedge \ x_1 \neq x_2 \wedge \ read(A, y) = v_1 \ \wedge v_1 \neq read(A, y)$$

# Deciding $\mathcal{T}_A$: Congruence Closure by Example

Question: Is formula $F$ given below $\mathcal{T}_A$-satisfiable?

$$x_1 = y \ \wedge \ x_1 \neq x_2 \ \wedge \ read(A, y) = v_1 \ \wedge$$
$$read(write(write(A, x1, v1), x2, v2), y) \neq read(A, y)$$

Use (read-over-write-1), and get: `unsatisfiable`

Use (read-over-write-2), and get:

$$F_2 : x_2 \neq y \wedge x_1 = y \wedge x_1 \neq x_2 \wedge read(A, y) = v_1 \wedge read(write(A, x_1, v_1), y) \neq read(A, y)$$

Use (read-over-write-1), and get: `unsatisfiable`

$$F_2' : x_1 = y \wedge \ x_2 \neq y \wedge \ x_1 = y \wedge \ x_1 \neq x_2 \wedge \ read(A, y) = v_1 \ \wedge v_1 \neq read(A, y)$$

# Deciding $\mathcal{T}_A$: Congruence Closure by Example

Question: Is formula $F$ given below $\mathcal{T}_A$-satisfiable?

$$x_1 = y \ \wedge \ x_1 \neq x_2 \ \wedge \ read(A, y) = v_1 \ \wedge$$
$$read(write(write(A, x1, v1), x2, v2), y) \neq read(A, y)$$

Use (read-over-write-1), and get: `unsatisfiable`

Use (read-over-write-2), and get:

$$F_2 : x_2 \neq y \wedge x_1 = y \wedge x_1 \neq x_2 \wedge read(A, y) = v_1 \wedge read(write(A, x_1, v_1), y) \neq read(A, y)$$

Use (read-over-write-1), and get: `unsatisfiable`

Use (read-over-write-2), and get:

# Deciding $\mathcal{T}_A$: Congruence Closure by Example

Question: Is formula $F$ given below $\mathcal{T}_A$-satisfiable?

$$x_1 = y \ \wedge \ x_1 \neq x_2 \ \wedge \ read(A, y) = v_1 \ \wedge$$
$$read(write(write(A, x1, v1), x2, v2), y) \neq read(A, y)$$

Use (read-over-write-1), and get: `unsatisfiable`

Use (read-over-write-2), and get:

$$F_2 : x_2 \neq y \wedge x_1 = y \wedge x_1 \neq x_2 \wedge read(A, y) = v_1 \wedge read(write(A, x_1, v_1), y) \neq read(A, y)$$

Use (read-over-write-1), and get: `unsatisfiable`

Use (read-over-write-2), and get:

$$F_2'' : x_1 \neq y \wedge x_2 \neq y \wedge x_1 = y \wedge x_1 \neq x_2 \wedge read(A, y) = v_1 \wedge read(A, y) \neq read(A, y)$$

# Deciding $\mathcal{T}_A$: Congruence Closure by Example

Question: Is formula $F$ given below $\mathcal{T}_A$-satisfiable?

$$x_1 = y \ \wedge \ x_1 \neq x_2 \ \wedge \ read(A, y) = v_1 \ \wedge$$
$$read(write(write(A, x1, v1), x2, v2), y) \neq read(A, y)$$

Use (read-over-write-1), and get: `unsatisfiable`

Use (read-over-write-2), and get:

$$F_2 : x_2 \neq y \wedge x_1 = y \wedge x_1 \neq x_2 \wedge read(A, y) = v_1 \wedge read(write(A, x_1, v_1), y) \neq read(A, y)$$

Use (read-over-write-1), and get: `unsatisfiable`

Use (read-over-write-2), and get: `unsatisfiable`

$$F_2'' : x_1 \neq y \wedge x_2 \neq y \wedge x_1 = y \wedge x_1 \neq x_2 \wedge read(A, y) = v_1 \wedge read(A, y) \neq read(A, y)$$

# Deciding $\mathcal{T}_A$: Congruence Closure by Example

Question: Is formula $F$ given below $\mathcal{T}_A$-satisfiable?

$$x_1 = y \ \wedge \ x_1 \neq x_2 \ \wedge \ read(A, y) = v_1 \ \wedge$$
$$read(write(write(A, x1, v1), x2, v2), y) \neq read(A, y)$$

Use (read-over-write-1), and get: `unsatisfiable`

Use (read-over-write-2), and get: `unsatisfiable`

Use (read-over-write-1), and get: `unsatisfiable`

Use (read-over-write-2), and get: `unsatisfiable`

$F$ is thus $\mathcal{T}_A$-unsatisfiable.

# Deciding $\mathcal{T}_A$: Congruence Closure Algorithm

Summary: $\mathcal{T}_A$-satisfiability of a formula $F$ is reduced to $\mathcal{T}_E$-satisfiability

Idea:

1. $F$ contains no *write*-terms. Then, treat *read*-terms as uninterpreted function terms
2. $F$ contains *write*-terms. Then, *write*-terms occur in the context of a *read*-term, so use (read-over-write) axioms to "eliminate" *write*-terms

# Deciding $\mathcal{T}_A$: Congruence Closure Algorithm

Summary: $\mathcal{T}_A$-satisfiability of a formula $F$ is reduced to $\mathcal{T}_E$-satisfiability

Idea:

1. $F$ contains no *write*-terms. Then, treat *read*-terms as uninterpreted function terms
2. $F$ contains *write*-terms. Then, *write*-terms occur in the context of a *read*-term, so use (read-over-write) axioms to "eliminate" *write*-terms

Computing $\mathcal{T}_A$-satisfiability is NP-complete.

# SMT Related Problems – Where are we now?

- ✓ Deciding theory: Check satisfiability of a set of *literals* in $\mathcal{T}_E$, $\mathcal{T}_A$

- ✓ Put together theory reasoning and SAT solving

# SMT Related Problems – Where are we now?

- ✓ Deciding theory: Check satisfiability of a set of *literals* in $\mathcal{T}_E$, $\mathcal{T}_A$

  What about in $\mathcal{T}_Q$? – visit linear optimization courses on the Simplex method

- ✓ Put together theory reasoning and SAT solving

- ? What about combination of theories: Given decision procedures for theories, how can we build a decision procedure for formulas using several theories?

# SMT Related Problems – Where are we now?

- ✓ Deciding theory: Check satisfiability of a set of *literals* in $\mathcal{T}_E$, $\mathcal{T}_A$

  What about in $\mathcal{T}_Q$? – visit linear optimization courses on the Simplex method

- ✓ Put together theory reasoning and SAT solving
- ? What about combination of theories: Given decision procedures for theories, how can we build a decision procedure for formulas using several theories?

**We next study satisfiability of formulas in combination of theories!**

# Outline

# Reasoning with a union/combination of theories

Given:

- Theories $T_1, \ldots, T_n$ in disjoint signatures $\Sigma_1, \ldots, \Sigma_n$;
- Quantifier-free formula $F$ in $\Sigma_1 \cup \ldots \cup \Sigma_n$.

# Reasoning with a union/combination of theories

Given:

- Theories $T_1, \ldots, T_n$ in disjoint signatures $\Sigma_1, \ldots, \Sigma_n$;
- Quantifier-free formula $F$ in $\Sigma_1 \cup \ldots \cup \Sigma_n$.

Question:

- Is $F$ satisfiable in $T_1 \cup \ldots \cup T_n$?

# Example

Is this set of formulas satisfiable?

$$x + 2 = y$$
$$f(read(write(a, x, 3), y - 2)) \neq f(y - x + 1)$$

# Example

Is this set of formulas satisfiable?

$$x + 2 = y$$
$$f(read(write(a, x, 3), y - 2)) \neq f(y - x + 1)$$

- linear arithmetic

# Example

Is this set of formulas satisfiable?

$$x + 2 = y$$
$$f(\mathit{read}(\mathit{write}(a, x, 3), y - 2)) \neq f(y - x + 1)$$

- linear arithmetic
- arrays

# Example

Is this set of formulas satisfiable?

$$x + 2 = y$$
$$f(read(write(a, x, 3), y - 2)) \neq f(y - x + 1)$$

- linear arithmetic
- arrays
- uninterpreted functions

# Example

Is this set of formulas satisfiable?

$$x + 2 = y$$
$$f(read(write(a, x, 3), y - 2)) \neq f(y - x + 1)$$

- linear arithmetic
- arrays
- uninterpreted functions

Sorts/types?

# Example

Is this set of formulas satisfiable?

$$x + 2 = y$$
$$f(read(write(a, x, 3), y - 2)) \neq f(y - x + 1)$$

- linear arithmetic
- arrays
- uninterpreted functions

Sorts/types?

- $x, y : int$

# Example

Is this set of formulas satisfiable?

$$x + 2 = y$$
$$f(read(write(a, x, 3), y - 2)) \neq f(y - x + 1)$$

- linear arithmetic
- arrays
- uninterpreted functions

Sorts/types?

- $x, y : int$
- $a$ : array[int,int]

# Example

Is this set of formulas satisfiable?

$$x + 2 = y$$
$$f(read(write(a, x, 3), y - 2)) \neq f(y - x + 1)$$

- linear arithmetic
- arrays
- uninterpreted functions

Sorts/types?

- $x, y : int$
- $a : \text{array[int,int]}$
- $f : int \rightarrow s$

# How to solve?

- Use DPLL($T$). This allows one to reduce the satisfiability problem for quantifier-free formulas to the satisfiability problem for conjunctions of literals.

# How to solve?

- Use DPLL($T$). This allows one to reduce the satisfiability problem for quantifier-free formulas to the satisfiability problem for conjunctions of literals.

Checking satisfiability of a conjunction of literals in a combination of theories?

# How to solve?

- Use DPLL($T$). This allows one to reduce the satisfiability problem for quantifier-free formulas to the satisfiability problem for conjunctions of literals.

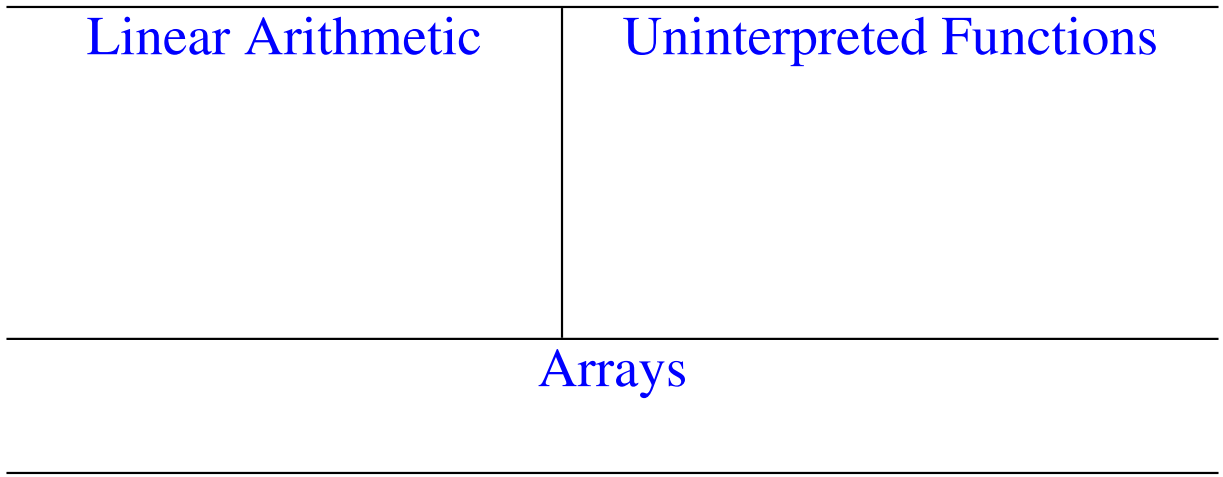
Checking satisfiability of a conjunction of literals in a combination of theories?

Idea:

- Separate reasoning in various theories
- Make reasoners exchange equalities

# Separating Reasoning

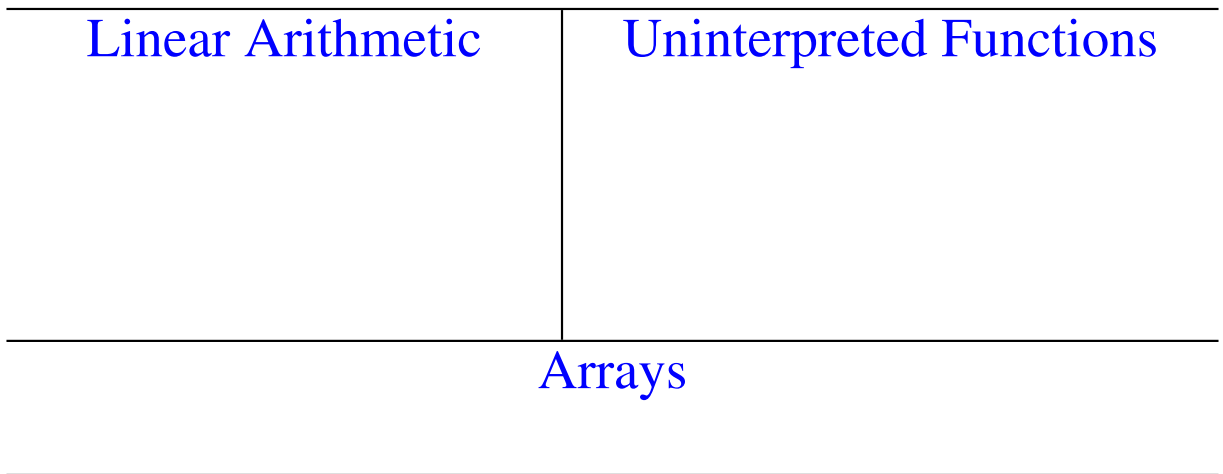
$$x + 2 = y$$
$$f(read(write(a, x, 3), y - 2)) \neq f(y - x + 1)$$



# Separating Reasoning

$x + 2 = y$

$f(read(write(a, x, 3), y - 2)) \neq f(y - x + 1)$



# Separating Reasoning

$$f(read(write(a, x, 3), y - 2)) \neq f(y - x + 1)$$

| Linear Arithmetic | Uninterpreted Functions |
| --- | --- |
| $y = x + 2$ | |
| Arrays | |

# Separating Reasoning

$$f(read(write(a, x, 3), y - 2)) \neq f(y - x + 1)$$

| Linear Arithmetic | Uninterpreted Functions |
| --- | --- |
| $y = x + 2$ | |
| Arrays | |

# Separating Reasoning

$$f(read(write(a, x, 3), y - 2)) \neq f(c_1)$$

| Linear Arithmetic | Uninterpreted Functions |
|---|---|
| $y = x + 2$<br>$c_1 = y - x + 1$ | |
| Arrays | |

# Separating Reasoning

$$f(read(write(a, x, 3), y - 2)) \neq f(c_1)$$

| Linear Arithmetic | Uninterpreted Functions |
|---|---|
| $y = x + 2$<br>$c_1 = y - x + 1$ | |
| Arrays | |

# Separating Reasoning

$$f(read(write(a, x, 3), c_2)) \neq f(c_1)$$

| Linear Arithmetic | Uninterpreted Functions |
|---|---|
| $y = x + 2$ | |
| $c_1 = y - x + 1$ | |
| $c_2 = y - 2$ | |
| Arrays | |

# Separating Reasoning

$$f(read(write(a, x, 3), c_2)) \neq f(c_1)$$

| Linear Arithmetic | Uninterpreted Functions |
|---|---|
| $y = x + 2$ | |
| $c_1 = y - x + 1$ | |
| $c_2 = y - 2$ | |
| $c_3 = 3$ | |
| Arrays | |

# Separating Reasoning

$$f(read(write(a, x, c_3), c_2)) \neq f(c_1)$$

| Linear Arithmetic | Uninterpreted Functions |
|---|---|
| $y = x + 2$ | |
| $c_1 = y - x + 1$ | |
| $c_2 = y - 2$ | |
| $c_3 = 3$ | |

Arrays

$$c_4 = read(write(a, x, c_3), c_2)$$

# Separating Reasoning

| Linear Arithmetic | Uninterpreted Functions |
|---|---|
| $y = x + 2$<br>$c_1 = y - x + 1$<br>$c_2 = y - 2$<br>$c_3 = 3$ | $f(c_4) \neq f(c_1)$ |
| Arrays | |
| $c_4 = read(write(a, x, c_3), c_2)$ | |

# Separating Reasoning

| Linear Arithmetic | Uninterpreted Functions |
|---|---|
| $y = x + 2$ | $f(c_4) \neq f(c_1)$ |
| $c_1 = y - x + 1$ | |
| $c_2 = y - 2$ | |
| $c_3 = 3$ | |
| **Arrays** | |
| $c_4 = read(write(a, x, c_3), c_2)$ | |

- Each step obviously preserves satisfiability; moreover every model of the modified set of literals is also a model of the original set.

# Separating Reasoning

**procedure** *SeparatingReasoning*($F$)
**input**: formula $F \in \mathcal{T}_1 \cup \cdots \cup \mathcal{T}_n$
**output**: formulas $F_1 \in \mathcal{T}_1, \ldots, F_n \in \mathcal{T}_n$ s.t.
$F$ and $F_1 \wedge \cdots \wedge F_n$ are equisatisfiable
**assumptions**: theory signatures $\Sigma_1, \ldots, \Sigma_n$ are disjoint
**parameters**: function $head(t)$ returning the root symbol of a term $t$
**begin**
**repeat as long as possible**
**if** $f \in \Sigma_i$ and $head(t) \in \Sigma_j$ with $i \neq j$:
rewrite $F[f(t_1, \ldots, t, \ldots, t_m)]$ into $F[f(t_1, \ldots, c, \ldots, t_m)] \wedge c = t$,
where $c$ is a fresh new variable
**if** $p \in \Sigma_i$ and $head(t) \in \Sigma_j$ with $i \neq j$:
rewrite $F[p(t_1, \ldots, t, \ldots, t_m)]$ into $F[p(t_1, \ldots, c, \ldots, t_m)] \wedge c = t$,
where $c$ is a fresh new variable
**if** $head(s) \in \Sigma_i$ and $head(t) \in \Sigma_j$ with $i \neq j$:
rewrite $F[s = t]$ into $F[s = c] \wedge c = t$,
where $c$ is a fresh new variable
**end repeat**
**return** modified $F$ as $F_1 \wedge \cdots \wedge F_n$, with each $F_i \in \mathcal{T}_i$

# Separating Reasoning

$$x + 2 = y$$
$$f(read(write(a, x, 3), y - 2)) \neq f(y - x + 1)$$

| Linear Arithmetic | Uninterpreted Functions |
|---|---|
| $y = x + 2$ | $f(c_4) \neq f(c_1)$ |
| $c_1 = y - x + 1$ | |
| $c_2 = y - 2$ | |
| $c_3 = 3$ | |
| **Arrays** | |
| $c_4 = read(write(a, x, c_3), c_2)$ | |

- Each step obviously preserves satisfiability; moreover every model of the modified set of literals is also a model of the original set.

# Separating Reasoning

$$x + 2 = y$$
$$f(read(write(a, x, 3), y - 2)) \neq f(y - x + 1)$$

| Linear Arithmetic | Uninterpreted Functions |
|---|---|
| $y = x + 2$<br>$c_1 = y - x + 1$<br>$c_2 = y - 2$<br>$c_3 = 3$ | $f(c_4) \neq f(c_1)$ |
| Arrays | |
| $c_4 = read(write(a, x, c_3), c_2)$ | |

- Each step obviously preserves satisfiability; moreover every model of the modified set of literals is also a model of the original set.
- In this example every "separated" subset of literals is satisfiable.

# Separating Reasoning

$$x + 2 = y$$
$$f(read(write(a, x, 3), y - 2)) \neq f(y - x + 1)$$

| Linear Arithmetic | Uninterpreted Functions |
|---|---|
| $y = x + 2$<br>$c_1 = y - x + 1$<br>$c_2 = y - 2$<br>$c_3 = 3$ | $f(c_4) \neq f(c_1)$ |
| Arrays | |
| $c_4 = read(write(a, x, c_3), c_2)$ | |

- Each step obviously preserves satisfiability; moreover every model of the modified set of literals is also a model of the original set.
- In this example every "separated" subset of literals is satisfiable.
- But this does not mean that the set of all literals is satisfiable.

# System to Solve

| Linear Arithmetic | Uninterpreted Functions |
| --- | --- |
| $y = x + 2$ | $f(c_4) \neq f(c_1)$ |
| $c_1 = y - x + 1$ | |
| $c_2 = y - 2$ | |
| $c_3 = 3$ | |
| Arrays | |
| $c_4 = read(write(a, x, c_3), c_2)$ | |

# Exchanging Equalities

| Linear Arithmetic | Uninterpreted Functions |
|---|---|
| $y = x + 2$<br>$c_1 = y - x + 1$<br>$c_2 = y - 2$<br>$c_3 = 3$ | $f(c_4) \neq f(c_1)$ |
| Arrays | Equalities |
| $c_4 = read(write(a, x, c_3), c_2)$ | |

# Exchanging Equalities

| Linear Arithmetic | Uninterpreted Functions |
|---|---|
| $y = x + 2$<br>$c_1 = y - x + 1$<br>$c_2 = y - 2$<br>$c_3 = 3$ | $f(c_4) \neq f(c_1)$ |
| Arrays | Equalities |
| $c_4 = read(write(a, x, c_3), c_2)$ | |

# Exchanging Equalities

| Linear Arithmetic | Uninterpreted Functions |
|---|---|
| $y = x + 2$<br>$c_1 = y - x + 1$<br>$c_2 = y - 2$<br>$c_3 = 3$ | $f(c_4) \neq f(c_1)$ |
| Arrays | Equalities |
| $c_4 = read(write(a, x, c_3), c_2)$ | $c_2 = x$ |

# Exchanging Equalities

| Linear Arithmetic | Uninterpreted Functions |
|---|---|
| $y = x + 2$<br>$c_1 = y - x + 1$<br>$c_2 = y - 2$<br>$c_3 = 3$ | $f(c_4) \neq f(c_1)$ |
| Arrays | Equalities |
| $c_4 = read(write(a, x, c_3), c_2)$ | $c_2 = x$ |

# Exchanging Equalities

| Linear Arithmetic | Uninterpreted Functions |
|---|---|
| $y = x + 2$<br>$c_1 = y - x + 1$<br>$c_2 = y - 2$<br>$c_3 = 3$ | $f(c_4) \neq f(c_1)$ |
| Arrays | Equalities |
| $c_4 = read(write(a, x, c_3), c_2)$ | $c_2 = x$ |

# Exchanging Equalities

| Linear Arithmetic | Uninterpreted Functions |
|---|---|
| $y = x + 2$<br>$c_1 = y - x + 1$<br>$c_2 = y - 2$<br>$c_3 = 3$ | $f(c_4) \neq f(c_1)$ |
| Arrays | Equalities |
| $c_4 = read(write(a, x, c_3), c_2)$ | $c_2 = x$<br>$c_4 = c_3$ |

# Exchanging Equalities

| Linear Arithmetic | Uninterpreted Functions |
|---|---|
| $y = x + 2$<br>$c_1 = y - x + 1$<br>$c_2 = y - 2$<br>$c_3 = 3$ | $f(c_4) \neq f(c_1)$ |
| Arrays | Equalities |
| $c_4 = read(write(a, x, c_3), c_2)$ | $c_2 = x$<br>$c_4 = c_3$ |

# Exchanging Equalities

| Linear Arithmetic | Uninterpreted Functions |
|---|---|
| $y = x + 2$<br>$c_1 = y - x + 1$<br>$c_2 = y - 2$<br>$c_3 = 3$ | $f(c_4) \neq f(c_1)$ |
| Arrays | Equalities |
| $c_4 = read(write(a, x, c_3), c_2)$ | $c_2 = x$<br>$c_4 = c_3$ |

# Exchanging Equalities

| Linear Arithmetic | Uninterpreted Functions |
|---|---|
| $y = x + 2$<br>$c_1 = y - x + 1$<br>$c_2 = y - 2$<br>$c_3 = 3$ | $f(c_4) \neq f(c_1)$ |
| **Arrays** | **Equalities** |
| $c_4 = read(write(a, x, c_3), c_2)$ | $c_2 = x$<br>$c_4 = c_3$<br>$c_1 = c_3$ |

# Exchanging Equalities

| Linear Arithmetic | Uninterpreted Functions |
|---|---|
| $y = x + 2$<br>$c_1 = y - x + 1$<br>$c_2 = y - 2$<br>$c_3 = 3$ | $f(c_4) \neq f(c_1)$ |
| Arrays | Equalities |
| $c_4 = read(write(a, x, c_3), c_2)$ | $c_2 = x$<br>$c_4 = c_3$<br>$c_1 = c_3$ |

# Exchanging Equalities

| Linear Arithmetic | Uninterpreted Functions |
|---|---|
| $y = x + 2$<br>$c_1 = y - x + 1$<br>$c_2 = y - 2$<br>$c_3 = 3$ | $f(c_4) \neq f(c_1)$ |
| Arrays | Equalities |
| $c_4 = read(write(a, x, c_3), c_2)$ | $c_2 = x$<br>$c_4 = c_3$<br>$c_1 = c_3$ |

# Exchanging Equalities

| Linear Arithmetic | Uninterpreted Functions |
|---|---|
| $y = x + 2$<br>$c_1 = y - x + 1$<br>$c_2 = y - 2$<br>$c_3 = 3$ | $f(c_4) \neq f(c_1)$ |
| **Arrays** | **Equalities** |
| $c_4 = read(write(a, x, c_3), c_2)$ | $c_2 = x$<br>$c_4 = c_3$<br>$c_1 = c_3$ |

- Congruence closure returns “unsatisfiable”.

# Exchanging Equalities

| Linear Arithmetic | Uninterpreted Functions |
|---|---|
| $y = x + 2$<br>$c_1 = y - x + 1$<br>$c_2 = y - 2$<br>$c_3 = 3$ | $f(c_4) \neq f(c_1)$ |
| **Arrays** | **Equalities** |
| $c_4 = read(write(a, x, c_3), c_2)$ | $c_2 = x$<br>$c_4 = c_3$<br>$c_1 = c_3$ |

- Congruence closure returns “unsatisfiable”.
- All derived equalities are implied by the initial set of literals. Therefore, the initial set is unsatisfiable.

# Exchanging Equalities

| Linear Arithmetic | Uninterpreted Functions |
|---|---|
| $y = x + 2$<br>$c_1 = y - x + 1$<br>$c_2 = y - 2$<br>$c_3 = 3$ | $f(c_4) \neq f(c_1)$ |
| **Arrays** | **Equalities** |
| $c_4 = read(write(a, x, c_3), c_2)$ | $c_2 = x$<br>$c_4 = c_3$<br>$c_1 = c_3$ |

- Congruence closure returns “unsatisfiable”.
- All derived equalities are implied by the initial set of literals. Therefore, the initial set is unsatisfiable.
- Therefore the original set (without extra variables) is unsatisfiable, too.

# Exchanging Equalities

| Linear Arithmetic | Uninterpreted Functions |
|---|---|
| $y = x + 2$<br>$c_1 = y - x + 1$<br>$c_2 = y - 2$<br>$c_3 = 3$ | $f(c_4) \neq f(c_1)$ |
| Arrays | Equalities |
| $c_4 = read(write(a, x, c_3), c_2)$ | $c_2 = x$<br>$c_4 = c_3$<br>$c_1 = c_3$ |

- Congruence closure returns "unsatisfiable".
- All derived equalities are implied by the initial set of literals. Therefore, the initial set is unsatisfiable.
- Therefore the original set (without extra variables) is unsatisfiable, too.

Not all theories can be combined (easily) in this way!