Automated Reasoning and Program Verification

Laura Kovács
TU Vienna
SMT questions - where are we now?

- Deciding a conjunction of literals in a theory: How can we check whether a set of literals is satisfiable, where a literal is a positive or negative atomic formula?

- Deciding arbitrary formulas in a theory: How can we put theory reasoning and SAT solving together? For example, for deciding an arbitrary formula in $T\mathcal{E}$, $T\mathcal{A}$ and $T\mathcal{Q}$?

- Combination of theories: Given decision procedures for theories, how can we build a decision procedure for formulas using several theories? For example, for deciding a formula in the $T\mathcal{E} \cup T\mathcal{A} \cup T\mathcal{Q}$?
SMT questions - where are we now?

▶ Deciding a conjunction of literals in a theory: How can we check whether a set of literals is satisfiable, where a literal is a positive or negative atomic formula?

We already have a decision procedure for $\mathcal{TE}$ and $\mathcal{TQ}$.
SMT questions - where are we now?

- **Deciding a conjunction of literals in a theory:** How can we check whether a set of *literals* is satisfiable, where a literal is a positive or negative atomic formula?

  We already have a decision procedure for $\mathcal{T}_E$ and $\mathcal{T}_Q$. What about $\mathcal{T}_A$?
SMT questions - where are we now?

- **Deciding a conjunction of literals in a theory:** How can we check whether a set of *literals* is satisfiable, where a literal is a positive or negative atomic formula?

  We already have a decision procedure for $T_E$ and $T_Q$. What about $T_A$?

- **Deciding arbitrary formulas in a theory:** How can we put theory reasoning and SAT solving together?
SMT questions - where are we now?

- **Deciding a conjunction of literals in a theory**: How can we check whether a set of *literals* is satisfiable, where a literal is a positive or negative atomic formula?

  We already have a decision procedure for $\mathcal{T}_E$ and $\mathcal{T}_Q$. What about $\mathcal{T}_A$?

- **Deciding arbitrary formulas in a theory**: How can we put theory reasoning and SAT solving together?

  For example, for deciding an arbitrary formula in $\mathcal{T}_E$, $\mathcal{T}_A$ and $\mathcal{T}_Q$?
SMT questions - where are we now?

- **Deciding a conjunction of literals in a theory**: How can we check whether a set of *literals* is satisfiable, where a literal is a positive or negative atomic formula?
  
  We already have a decision procedure for $T_E$ and $T_Q$. What about $T_A$?

- **Deciding arbitrary formulas in a theory**: How can we put theory reasoning and SAT solving together?
  
  For example, for deciding an arbitrary formula in $T_E$, $T_A$ and $T_Q$?

- **Combination of theories**: Given decision procedures for theories, how can we build a decision procedure for formulas using several theories?
SMT questions - where are we now?

- Deciding a conjunction of literals in a theory: How can we check whether a set of literals is satisfiable, where a literal is a positive or negative atomic formula?
  We already have a decision procedure for $T_E$ and $T_Q$. What about $T_A$?

- Deciding arbitrary formulas in a theory: How can we put theory reasoning and SAT solving together?
  For example, for deciding an arbitrary formula in $T_E$, $T_A$ and $T_Q$?

- Combination of theories: Given decision procedures for theories, how can we build a decision procedure for formulas using several theories?
  For example, for deciding a formula in the $T_E \cup T_A \cup T_Q$?
Outline

Theory of Arrays

SMT and Non-Unit Clauses
Deciding $\mathcal{T}_A$

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

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

  - $\text{read}$ is a binary function: $\text{read}(A, x)$ is the value of array $A$ at position $x$
  - $\text{write}$ is a ternary function: $\text{write}(A, x, v)$ is the modified array $A$ in which the element at position $x$ has value $v$

From now on, we consider $\mathcal{T}_A$ as part of the language.

- the following axioms:

  - equality axioms
    - $x = y \rightarrow \text{read}(\text{write}(A, x, v), y) = v$ (read-over-write 1)
    - $x \neq y \rightarrow \text{read}(\text{write}(A, x, v), y) = \text{read}(A, y)$ (read-over-write 2)

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

Idea:
1. $F$ contains no $\text{write}$-terms. Then, treat $\text{read}$-terms as uninterpreted function terms
2. $F$ contains $\text{write}$-terms. Then, $\text{write}$-terms occur in the context of a $\text{read}$-term, so use (read-over-write) axioms to "eliminate" $\text{write}$-terms
Deciding $\mathcal{T}_A$

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

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

  * $\text{read}$ is a binary function: $\text{read}(A, x)$ is the value of array $A$ at position $x$
  * $\text{write}$ is a ternary function: $\text{write}(A, x, v)$ 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 $\Sigma_A = \{\text{read}, \text{write}\}$ where

  - $\text{read}$ is a binary function: $\text{read}(A, x)$ is the value of array $A$ at position $x$
  - $\text{write}$ is a ternary function: $\text{write}(A, x, v)$ 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 $\mathcal{T}_E$*

    \[
    \begin{align*}
    x = y & \Rightarrow \text{read}(\text{write}(A, x, v), y) = v \quad \text{(read-over-write 1)} \\
    x \neq y & \Rightarrow \text{read}(\text{write}(A, x, v), y) = \text{read}(A, y) \quad \text{(read-over-write 2)}
    \end{align*}
    \]
Deciding $\mathcal{T}_A$: Congruence Closure Algorithm

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

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

  - $\text{read}$ is a binary function: $\text{read}(A, x)$ is the value of array $A$ at position $x$
  - $\text{write}$ is a ternary function: $\text{write}(A, x, v)$ 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 $\mathcal{T}_E$
    
    $\begin{align*}
    x = y & \rightarrow \text{read}(\text{write}(A, x, v), y) = v \quad \text{(read-over-write 1)} \\
    x \neq y & \rightarrow \text{read}(\text{write}(A, x, v), y) = \text{read}(A, y) \quad \text{(read-over-write 2)}
    \end{align*}$

  $\mathcal{T}_A$-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 $\Sigma_A = \{\text{read}, \text{write}\}$ where

  - $\text{read}$ is a binary function: $\text{read}(A, x)$ is the value of array $A$ at position $x$
  - $\text{write}$ is a ternary function: $\text{write}(A, x, v)$ 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 $\mathcal{T}_E$
    
    $x = y \rightarrow \text{read}(\text{write}(A, x, v), y) = v$  \hspace{1cm} (read-over-write 1)
    
    $x \neq y \rightarrow \text{read}(\text{write}(A, x, v), y) = \text{read}(A, y)$  \hspace{1cm} (read-over-write 2)

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

Idea:

1. $F$ contains no $\text{write}$-terms. Then, treat $\text{read}$-terms as uninterpreted function terms

2. $F$ contains $\text{write}$-terms. Then, $\text{write}$-terms occur in the context of a $\text{read}$-term, so use (read-over-write) axioms to “eliminate” $\text{write}$-terms
Deciding $\mathcal{T}_A$: Congruence Closure Algorithm

Algorithm for deciding $\mathcal{T}_A$

1. $F$ contains no $\text{write}$-terms:
   - use fresh function symbol $f_A$ for array variables $A$
   - replace $\text{read}(A, x)$ 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 $F_1$:
     \[
     F_1 : x = y \land F[v]
     \]
     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 $\text{read}(A, x)$ with $f_A(x)$ in $F$
   - decide and return $\mathcal{T}_E$-satisfiability of resulting formula

2. $F$ contains write-terms, say $\text{read} (\text{write}(A, x, v), y)$
   - Using (read-over-write 1), replace $F$ by the following formula $F_1$:
     \[
     F_1 : x = y \land F[v]
     \]
     where $F[v]$ denotes the formula obtained by replacing $\text{read}(\text{write}(A, x, v), y)$ 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 $\text{read}(A, x)$ with $f_A(x)$ in $F$
   - decide and return $\mathcal{T}_E$-satisfiability of resulting formula

2. $F$ contains write-terms, say $\text{read}(\text{write}(A, x, v), y)$
   - Using (read-over-write 1), replace $F$ by the following formula $F_1$:
     \[
     F_1 : x = y \land F[v]
     \]
     where $F[v]$ denotes the formula obtained by replacing $\text{read}(\text{write}(A, x, v), y)$ 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 $F_2$:
     \[
     F_1 : x \neq y \land F[\text{read}(A, y)]
     \]
     where $F[\text{read}(A, y)]$ denotes the formula by replacing $\text{read}(\text{write}(A, x, v), y)$ with $\text{read}(A, y)$ in $F$. If $F_1$ and $F_2$ are $\mathcal{T}_A$-unsatisfiable, return unsatisfiable
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 $\text{read}(A, x)$ with $f_A(x)$ in $F$
   - decide and return $\mathcal{T}_E$-satisfiability of resulting formula

2. $F$ contains *write*-terms, say $\text{read}(\text{write}(A, x, v), y)$
   - Using (read-over-write 1), replace $F$ by the following formula $F_1$:
     $$F_1 : x = y \land F[v]$$
     where $F[v]$ denotes the formula obtained by replacing $\text{read}(\text{write}(A, x, v), y)$ 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 $F_2$:
     $$F_1 : x \neq y \land F[\text{read}(A, y)]$$
     where $F[\text{read}(A, y)]$ denotes the formula by replacing $\text{read}(\text{write}(A, x, v), y)$ with $\text{read}(A, y)$ 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 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 $F_1$:
     \[ F_1 : x = y \land F[v] \]
     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 $F_2$:
     \[ F_1 : x \neq y \land F[read(A, y)] \]
     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?

$$x_1 = y \land x_1 \neq x_2 \land \text{read}(A, y) = v_1 \land \text{read(write(write(A, x_1, v_1), x_2, v_2), y) \neq \text{read}(A, y)}$$
Deciding $\mathcal{T}_A$: Congruence Closure by Example

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

\[
x_1 = y \land x_1 \neq x_2 \land \text{read}(A, y) = v_1 \land \text{read(}\text{write(}\text{write}(A, x_1, v_1), x_2, v_2), y) \neq \text{read}(A, y)
\]

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

\[
F_1 : x_2 = y \land x_1 = y \land x_1 \neq x_2 \land \text{read}(A, y) = v_1 \land v_2 \neq \text{read}(A, y)
\]
Deciding $\mathcal{T}_A$: Congruence Closure by Example

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

\[
\begin{align*}
x_1 &= y \land x_1 \neq x_2 \land \text{read}(A, y) = v_1 \land \\
\text{read}(\text{write}(\text{write}(A, x_1, v_1), x_2, v_2), y) &\neq \text{read}(A, y)
\end{align*}
\]

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

\[
F_1 : x_2 = y \land x_1 = y \land x_1 \neq x_2 \land \text{read}(A, y) = v_1 \land v_2 \neq \text{read}(A, y)
\]

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

\[
F'_1 : x_2 = y \land x_1 = y \land x_1 \neq x_2 \land f_A(y) = v_1 \land v_2 \neq f_A(y)
\]
Deciding $\mathcal{T}_A$: Congruence Closure by Example

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

$$x_1 = y \land x_1 \neq x_2 \land \text{read}(A, y) = v_1 \land \text{read}(\text{write}(\text{write}(A, x_1, v_1), x_2, v_2), y) \neq \text{read}(A, y)$$

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

$$F_1 : x_2 = y \land x_1 = y \land x_1 \neq x_2 \land \text{read}(A, y) = v_1 \land v_2 \neq \text{read}(A, y)$$

$F_1$ contains no $\text{write}$-terms, so rewrite it to:

$$F'_1 : x_2 = y \land x_1 = y \land x_1 \neq x_2 \land f_A(y) = v_1 \land v_2 \neq f_A(y)$$

$F'_1$ is $\mathcal{T}_E$-unsatisfiable.
Deciding $\mathcal{T}_A$: Congruence Closure by Example

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

$$x_1 = y \land x_1 \neq x_2 \land read(A, y) = v_1 \land \neg read(write(write(A, x_1, v_1), x_2, v_2), y) = read(A, y)$$

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

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

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

$F'_1 : x_2 = y \land x_1 = y \land x_1 \neq x_2 \land f_A(y) = v_1 \land v_2 \neq f_A(y)$

$F'_1$ is $\mathcal{T}_E$-unsatisfiable.
Deciding $\mathcal{T}_A$: Congruence Closure by Example

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

$$x_1 = y \land x_1 \neq x_2 \land \text{read}(A, y) = v_1 \land \text{read(write(write(A, x_1, v_1), x_2, v_2), y)} \neq \text{read}(A, y)$$

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

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

$F_2 : x_2 \neq y \land x_1 = y \land x_1 \neq x_2 \land \text{read}(A, y) = v_1 \land \text{read(write(A, x_1, v_1), y)} \neq \text{read}(A, y)$
Deciding $\mathcal{T}_A$: Congruence Closure by Example

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

\[
x_1 = y \land x_1 \neq x_2 \land \text{read}(A, y) = v_1 \land \\
\text{read}(\text{write}(\text{write}(A, x_1, v_1), x_2, v_2), y) \neq \text{read}(A, y)
\]

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

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

$F_2 : x_2 \neq y \land x_1 = y \land x_1 \neq x_2 \land \text{read}(A, y) = v_1 \land \text{read}(\text{write}(A, x_1, v_1), y) \neq \text{read}(A, y)$

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?

$$x_1 = y \land x_1 \neq x_2 \land \text{read}(A, y) = v_1 \land \text{read}(\text{write}(\text{write}(A, x_1, v_1), x_2, v_2), y) \neq \text{read}(A, y)$$

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

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

$$F_2 : x_2 \neq y \land x_1 = y \land x_1 \neq x_2 \land \text{read}(A, y) = v_1 \land \text{read}(\text{write}(A, x_1, v_1), y) \neq \text{read}(A, y)$$

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

$$F'_2 : x_1 = y \land x_2 \neq y \land x_1 = y \land x_1 \neq x_2 \land \text{read}(A, y) = v_1 \land v_1 \neq \text{read}(A, y)$$
Deciding $\mathcal{T}_A$: Congruence Closure by Example

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

\[
x_1 = y \land x_1 \neq x_2 \land \text{read}(A, y) = v_1 \land \text{read(write(write(A, x_1, v_1), x_2, v_2), y) \neq read(A, y)}
\]

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

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

\[
F_2 : x_2 \neq y \land x_1 = y \land x_1 \neq x_2 \land \text{read}(A, y) = v_1 \land \text{read(write(A, x_1, v_1), y) \neq read(A, y)}
\]

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

\[
F'_2 : x_1 = y \land x_2 \neq y \land x_1 = y \land x_1 \neq x_2 \land \text{read}(A, y) = v_1 \land v_1 \neq \text{read}(A, y)
\]
Deciding $\mathcal{T}_A$: Congruence Closure by Example

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

$$x_1 = y \land x_1 \neq x_2 \land \text{read}(A, y) = v_1 \land \text{read(}\text{write(}\text{write(}A, x_1, v_1), x_2, v_2), y) \neq \text{read}(A, y)$$

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

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

$$F_2 : x_2 \neq y \land x_1 = y \land x_1 \neq x_2 \land \text{read}(A, y) = v_1 \land \text{read(}\text{write(}A, x_1, v_1), y) \neq \text{read}(A, y)$$

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?

$$x_1 = y \land x_1 \neq x_2 \land \text{read}(A, y) = v_1 \land \text{read(write(write(A, x_1, v_1), x_2, v_2), y)} \neq \text{read}(A, y)$$

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

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

$$F_2: x_2 \neq y \land x_1 = y \land x_1 \neq x_2 \land \text{read}(A, y) = v_1 \land \text{read(write(A, x_1, v_1), y)} \neq \text{read}(A, y)$$

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

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

$$F''_2: x_1 \neq y \land x_2 \neq y \land x_1 = y \land x_1 \neq x_2 \land \text{read}(A, y) = v_1 \land \text{read}(A, y) \neq \text{read}(A, y)$$
Deciding $\mathcal{T}_A$: Congruence Closure by Example

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

\[
x_1 = y \land x_1 \neq x_2 \land read(A, y) = v_1 \land read(write(write(A, x_1, v_1), x_2, v_2), y) \neq read(A, y)
\]

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

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

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

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

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

$F''_2 : x_1 \neq y \land x_2 \neq y \land x_1 = y \land x_1 \neq x_2 \land read(A, y) = v_1 \land read(A, y) \neq read(A, y)$
Deciding $\mathcal{T}_A$: Congruence Closure by Example

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

\[ x_1 = y \land x_1 \neq x_2 \land \text{read}(A, y) = v_1 \land \text{read(write(write(A, x_1, v_1), x_2, v_2), y)} \neq \text{read}(A, y) \]

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

\[ F \text{ is thus } \mathcal{T}_A\text{-unsatisfiable.} \]
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.
Problems – Where are we now?

✓ Deciding theory: Check satisfiability of a set of literals in $T_E$, $T_A$ and $T_Q$
Problems – Where are we now?

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

? What about deciding arbitrary formulas in a theory: How can we put together theory reasoning and SAT solving?
Problems — Where are we now?

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

? What about deciding arbitrary formulas in a theory: How can we 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?
Problems – Where are we now?

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

? What about deciding arbitrary formulas in a theory: How can we 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 arbitrary formulas in a theory.
Outline

Theory of Arrays

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

**Example:** Let’s try to prove the validity of the formula:

$$F : (a = b \lor a = c) \land f(a) = a \rightarrow f(f(b)) = a \lor f(c) = a.$$  

This is equivalent to establishing unsatisfiability of

- $a = b \lor a = c$
- $f(a) = a$
- $f(f(b)) \neq a$
- $f(c) \neq a$
Deciding $\mathcal{T}_E$: Problems Using Non-Unit Clauses

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

$$F : (a = b \lor a = c) \land f(a) = a \rightarrow f(f(b)) = a \lor f(c) = a.$$  

This is equivalent to establishing unsatisfiability of

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

We have a non-unit clause, so we can’t 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:

\[
\begin{align*}
&a = b \lor a = c \\
&f(a) = a \\
&f(f(b)) \neq a \\
&f(c) \neq a \\
\end{align*}
\]

\[
\begin{align*}
&\textit{p}_1 \lor \textit{p}_2 \\
&\textit{p}_3 \\
&\neg \textit{p}_4 \\
&\neg \textit{p}_5 \\
\end{align*}
\]

\[
\begin{align*}
&\textit{p}_1 : a = b \\
&\textit{p}_2 : a = c \\
&\textit{p}_3 : f(a) = a \\
&\textit{p}_4 : f(f(b)) = a \\
&\textit{p}_5 : f(c) = a \\
\end{align*}
\]
Idea: Use a SAT Solver

Add a propositional symbol to name every theory atom:

\[ a = b \lor a = c \]
\[ f(a) = a \]
\[ f(f(b)) \neq a \]
\[ f(c) \neq a \]

\[ p_1 \lor p_2 \]
\[ p_3 \]
\[ \neg p_4 \]
\[ \neg p_5 \]

\[ p_1 : a = b \]
\[ p_2 : a = c \]
\[ p_3 : f(a) = a \]
\[ p_4 : f(f(b)) = a \]
\[ p_5 : f(c) = a \]

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:

\[
\begin{align*}
\text{atom} &= a = b \lor a = c \\
f(a) &= a \\
f(f(b)) &\neq a \\
f(c) &\neq a
\end{align*}
\]
\[
\begin{align*}
p_1 \lor p_2 \\
p_3 \\
\neg p_4 \\
\neg p_5
\end{align*}
\]
\[
\begin{align*}
p_1 : a = b \\
p_2 : a = c \\
p_3 : f(a) = a \\
p_4 : f(f(b)) = a \\
p_5 : f(c) = a
\end{align*}
\]

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:

\[
\begin{align*}
  a = b \vee a = c & \quad \quad p_1 \lor p_2 \\
  f(a) = a & \quad \quad p_3 \\
  f(f(b)) \neq a & \quad \quad \neg p_4 \\
  f(c) \neq a & \quad \quad \neg p_5 \\
\end{align*}
\]

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 \( \neg L_1 \lor \ldots \lor \neg L_n \) 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:

\[
\begin{align*}
  a &= b \lor a = c \\
  f(a) &= a \\
  f(f(b)) &\neq a \\
  f(c) &\neq a
\end{align*}
\]

**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 \( \neg L_1 \lor \ldots \lor \neg L_n \) 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 $T_E$, $T_A$, and $T_Q$.

✓ Put together theory reasoning and SAT solving
Problems – Where are we now?

✓ Deciding theory: Check satisfiability of a set of literals in $\mathcal{T}_E$, $\mathcal{T}_A$ 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?
Problems – Where are we now?

✓ Deciding theory: Check satisfiability of a set of literals in $T_E$, $T_A$ and $T_Q$.

✓ 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?

We next study satisfiability of formulas in combination of theories!