Satisfiability Modulo Theories Solver
Decision procedure

Lecturer: Yu-Fang Chen

Institute of Information Science
Academia Sinica

Based on Reynolds(2017), Tsai(2017)
Thanks to Yi-Fan Lin for making the slides

Table of Contents

From SAT to SMT

- We've known the decision procedure for **SAT** problems.
  - DPLL algorithm
- What happened when it comes to **First-Order Logic**,

  e.g.

$$(x + y < 3 \lor x < 0) \land (\neg(x < 0) \lor x = y + 3) \land (y = 4),$$

  is this formula satisfiable under the theory of LIA?

  $\rightarrow$ We can apply **SMT** solver

Satisfiability Modulo Theories (SMT) solver

- Rely on **DPLL(T)** algorithm, an extension of DPLL, where **T** is a set of first-order theories.
- A **first-order theory** is defined by:
  - Signature(Σ): a set of non-logical symbols
  - Axioms(must be satisfied): a set of Σ-formula
- SAT solver operations: **Propagate**, **Decide** and **Backtrack**.

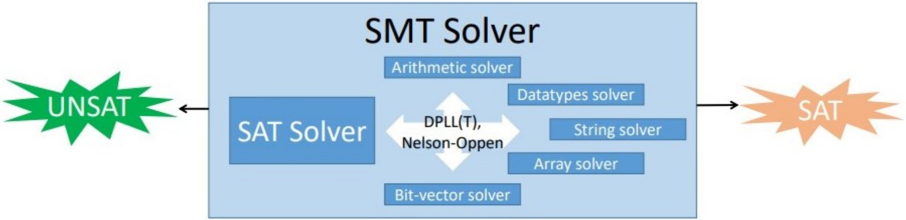

Figure 1: Basic architecture of a SMT solver [1]

DPLL(T) Algorithm - An Example

$$\phi := (x + y < 3 \ \vee \ x < 0) \wedge (\neg(x < 0) \ \vee \ x = y + 3) \wedge y = 4$$

$$\xrightarrow{\text{abstraction}} \phi_p := (a_0 \ \vee \ a_1) \wedge (\neg a_1 \ \vee \ a_2) \wedge a_3$$

DPLL(T) Algorithm - An Example

$$\phi := (x + y < 3 \lor x < 0) \land (\neg(x < 0) \lor x = y + 3) \land \boxed{y = 4}$$

$$\phi_p := (a_0 \lor a_1) \land (\neg a_1 \lor a_2) \land \boxed{a_3}$$

- Propagate: $a_3 \mapsto \mathrm{T}$

DPLL(T) Algorithm - An Example

$$\phi := (x + y < 3 \ \lor \ \boxed{x < 0}) \land (\boxed{\neg(x < 0)} \ \lor \ x = y + 3) \land \boxed{y = 4}$$

$$\phi_p := (a_0 \ \lor \ \boxed{a_1}) \land (\boxed{\neg a_1} \ \lor \ a_2) \land \boxed{a_3}$$

- Propagate: $a_3 \mapsto T$
- Decide: $a_1 \mapsto T$

DPLL(T) Algorithm - An Example

$$\phi := (x + y < 3 \ \lor \ x < 0) \land (\neg(x < 0) \ \lor \ x = y + 3) \land y = 4$$

$$\phi_p := (a_0 \ \lor \ a_1) \land (\neg a_1 \ \lor \ a_2) \land a_3$$

- Propagate: $a_3 \mapsto \mathrm{T}$
- Decide: $a_1 \mapsto \mathrm{T}$
- Propagate: $a_2 \mapsto \mathrm{T}$

DPLL(T) Algorithm - An Example

$$\phi := (x + y < 3 \lor x < 0) \land (\neg(x < 0) \lor x = y + 3) \land y = 4$$

$$\phi_p := (a_0 \lor a_1) \land (\neg a_1 \lor a_2) \land a_3$$

- Propagate: $a_3 \mapsto \mathrm{T}$
- Decide: $a_1 \mapsto \mathrm{T}$
- Propagate: $a_2 \mapsto \mathrm{T}$
- Pass assignment $\alpha := \{a_1 \mapsto \mathrm{T}, a_2 \mapsto \mathrm{T}, a_3 \mapsto \mathrm{T}\}$ to LIA solver, LIA solver solves $(y = 4 \land x < 0 \land x = y + 3)$ and gets **UNSAT**

DPLL(T) Algorithm - An Example

$$\phi := (x + y < 3 \;\vee\; x < 0) \wedge (\neg(x < 0) \;\vee\; x = y + 3) \wedge y = 4$$
$$\wedge (\neg(y = 4) \vee \neg(x < 0) \vee \neg(x = y + 3))$$

$$\phi_p := (a_0 \;\vee\; a_1) \wedge (\neg a_1 \;\vee\; a_2) \wedge a_3 \wedge (\neg a_3 \vee \neg a_1 \vee \neg a_2)$$

- Propagate: $a_3 \mapsto \mathrm{T}$
- Decide: $a_1 \mapsto \mathrm{T}$
- Propagate: $a_2 \mapsto \mathrm{T}$
- Pass assignment $\alpha := \{a_1 \mapsto \mathrm{T}, a_2 \mapsto \mathrm{T}, a_3 \mapsto \mathrm{T}\}$ to LIA solver, LIA solver solves $(y = 4 \wedge \neg(x < 0) \wedge x = y + 3)$ and gets **UNSAT**.
  $\Rightarrow$ Add blocking clause

DPLL(T) Algorithm - An Example

$$\phi := (x + y < 3 \lor x < 0) \land (\neg(x < 0) \lor x = y + 3) \land y = 4$$
$$\land (\neg(y = 4) \lor \neg(x < 0) \lor \neg(x = y + 3))$$

$$\phi_p := (a_0 \lor a_1) \land (\neg a_1 \lor a_2) \land a_3 \land (\neg a_3 \lor \neg a_1 \lor \neg a_2)$$

- Propagate: $a_3 \mapsto \mathsf{T}$
- Decide: $a_1 \mapsto \mathsf{T}$
- Propagate: $a_2 \mapsto \mathsf{T}$
- Pass assignment $\alpha := \{a_1 \mapsto \mathsf{T}, a_2 \mapsto \mathsf{T}, a_3 \mapsto \mathsf{T}\}$ to LIA solver, LIA solver solves $(y = 4 \land x < 0 \land x = y + 3)$ and gets **UNSAT**.
  $\Rightarrow$ Add blocking clause
- Conflict! backtrack to the decision

DPLL(T) Algorithm - An Example

$$\phi := (x + y < 3 \ \vee \ x < 0) \wedge (\neg(x < 0) \ \vee \ x = y + 3) \wedge y = 4$$
$$\wedge (\neg(y = 4) \vee \neg(x < 0) \vee \neg(x = y + 3))$$

$$\phi_p := (a_0 \ \vee \ a_1) \wedge (\neg a_1 \ \vee \ a_2) \wedge a_3 \wedge (\neg a_3 \vee \neg a_1 \vee \neg a_2)$$

- Propagate: $a_3 \mapsto$ T
- Backtrack: $a_1 \mapsto$ F

DPLL(T) Algorithm - An Example

$$\phi := (\, x+y<3 \;\vee\; x<0 \,) \wedge (\, \neg(x<0) \;\vee\; x=y+3 \,) \wedge y=4$$
$$\wedge\, (\, \neg(y=4) \vee \neg(x<0) \vee \neg(x=y+3) \,)$$

$$\phi_p := (\, a_0 \;\vee\; a_1 \,) \wedge (\, \neg a_1 \;\vee\; a_2 \,) \wedge a_3 \wedge (\, \neg a_3 \vee \neg a_1 \vee \neg a_2 \,)$$

- Propagate: $a_3 \mapsto \mathrm{T}$
- Backtrack: $a_1 \mapsto \mathrm{F}$
- Propagate: $a_0 \mapsto \mathrm{T}$

DPLL(T) Algorithm - An Example

$$\phi := (x + y < 3 \lor x < 0) \land (\neg(x < 0) \lor x = y + 3) \land y = 4$$
$$\land (\neg(y = 4) \lor \neg(x < 0) \lor \neg(x = y + 3))$$

$$\phi_p := (a_0 \lor a_1) \land (\neg a_1 \lor a_2) \land a_3 \land (\neg a_3 \lor \neg a_1 \lor \neg a_2)$$

- Propagate: $a_3 \mapsto \mathrm{T}$
- Backtrack: $a_1 \mapsto \mathrm{F}$
- Propagate: $a_0 \mapsto \mathrm{T}$
- Pass assignment $\alpha := \{a_0 \mapsto \mathrm{T}, a_1 \mapsto \mathrm{F}, a_3 \mapsto \mathrm{T}\}$ to LIA solver, LIA solver solves $(x + y < 3 \land \neg(x < 0) \land y = 4)$ and gets **UNSAT**.

DPLL(T) Algorithm - An Example

$$\phi := (x + y < 3 \lor x < 0) \land (\neg(x < 0) \lor x = y + 3) \land y = 4$$
$$\land (\neg(y = 4) \lor \neg(x < 0) \lor \neg(x = y + 3))$$
$$\land (\neg(x + y < 3) \lor (x < 0) \lor \neg(y = 4))$$

$$\phi_p := (a_0 \lor a_1) \land (\neg a_1 \lor a_2) \land a_3 \land (\neg a_3 \lor \neg a_1 \lor \neg a_2)$$
$$\land (\neg a_0 \lor a_1 \lor \neg a_3)$$

- Propagate: $a_3 \mapsto \mathrm{T}$
- Backtrack: $a_1 \mapsto \mathrm{F}$
- Propagate: $a_0 \mapsto \mathrm{T}$
- Pass assignment $\alpha := \{a_0 \mapsto \mathrm{T}, a_1 \mapsto \mathrm{F}, a_3 \mapsto \mathrm{T}\}$ to LIA solver, LIA solver solves $(x + y < 3 \land \neg(x < 0) \land y = 4)$ and gets **UNSAT**.
  $\Rightarrow$ Add blocking clause

DPLL(T) Algorithm - An Example

$$\phi := (x+y<3 \ \vee\ x<0) \wedge (\neg(x<0) \ \vee\ x=y+3) \wedge y=4$$
$$\wedge (\neg(y=4) \vee \neg(x<0) \vee \neg(x=y+3))$$
$$\wedge (\neg(x+y<3) \vee x<0 \vee \neg(y=4))$$

$$\phi_p := (a_0 \ \vee\ a_1) \wedge (\neg a_1 \ \vee\ a_2) \wedge a_3 \wedge (\neg a_3 \vee \neg a_1 \vee \neg a_2)$$
$$\wedge (\neg a_0 \vee a_1 \vee \neg a_3)$$

- Propagate: $a_3 \mapsto \mathrm{T}$
- Backtrack: $a_1 \mapsto \mathrm{F}$
- Propagate: $a_0 \mapsto \mathrm{T}$
- Pass assignment $\alpha := \{a_0 \mapsto \mathrm{T}, a_1 \mapsto \mathrm{F}, a_3 \mapsto \mathrm{T}\}$ to LIA solver, LIA solver solves $(x+y<3 \wedge \neg(x<0) \wedge y=4)$ and gets **UNSAT**.
- $\Rightarrow$ Add blocking clause. No decision to Backtrack, return **UNSAT**

Satisfiability Modulo Theories (SMT) solver

- Basic Idea:
  1. The SAT solver checks whether the **propositional abstraction** of the formula is satisfiable
     - If so, decide an assignment for **each literal**.
     - If not, backtrack. If backtracking is unavailable, return **UNSAT**(T-unsatisfiable).
  2.
     -
     -

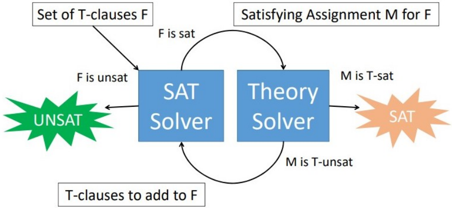

Figure 2: Interactions inside a SMT solver[1]

Satisfiability Modulo Theories (SMT) solver

- Basic Idea:
  1. The SAT solver checks whether the **propositional abstraction** of the formula is satisfiable.
     - If so, decide an assignment for **each literal**.
     - If not, backtrack. If backtracking is unavailable, return **UNSAT**(T-unsatisfiable).
  2. Then, the theory solver checks whether the assignment is satisfiable.
     - If so, return **SAT**(T-satisfiable).
     - If not, add blocking clauses to the formula, go back to step 1.

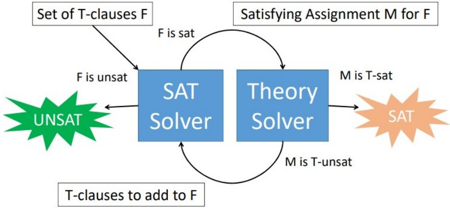

Figure 2: Interactions inside a SMT solver[1]

Propositional Abstraction - Exercise

Perform propositional abstraction:

- $F(a) = F(F(b)) \wedge a = 5 \wedge (\neg(b = 5) \vee F(b) = 5)$

- $(a[i] + 4 = 5$ $) \wedge (a[0] = a[j] \vee a[0] = a[i]) \wedge \neg(i = j))$

Propositional Abstraction - Exercise

Perform propositional abstraction:

- $F(a) = F(F(b)) \land a = 5 \land (\neg(b = 5) \lor F(b) = 5)$

  $\Rightarrow a_0 \land a_1 \land (\neg a_2 \lor a_3)$

- $(a[i] + 4 = 5 \lor ai \leftarrow x[j] < 0) \land (i = j \lor a[0] = a[i]) \land \neg(i = j))$

  $\Rightarrow (b_0 \lor b_1) \land (b_2 \lor b_3) \land \neg b_2$

Perform DPLL(LIA) Algorithm to solve the formula:
(you can omit the decision procedure of LIA solver)

- $(x > 0 \vee x + y < 1) \wedge (x + y = 2 \vee y = 5) \wedge (x > 3 \vee \neg(x + y = 2))$
- $(x < y \vee x = z \vee x + z > 7) \wedge x > 5 \wedge z = 4 \wedge y + z < 3$

Table of Contents

Theory solvers

- We've provided a walkthrough of DPLL(T) with an example, but one may ask: How do theory solvers work?
- Formally, a theory solver should(assuming ϕ is the input formula):
  - Return **SAT** only if ϕ is T-satisfiable.
  - Return **UNSAT** only if ϕ is T-unsatisfiable.
  - Terminate.
- In practice, a theory solver supports following features:
  - Return an **interpretation** when ϕ is T-satisfiable.
  - Return a **conflict clause** when ϕ is T-unsatisfiable.

Theory solvers

Here we focus on the decision procedure for the **quantifier-free fragment** of **first-order theories**.

- Equality and Uninterpreted Functions
- Arrays
- Linear Integer Arithmetic (Simplex method)
- Bit Vectors
- Recursive Datatypes
- ...

Table of Contents

Theory of EUF

Signature:

$$\Sigma := \{=, a, b, c, \ldots, A, B, C, \ldots\}$$

where $\{a, b, c, \ldots, A, B, C, \ldots\}$ are symbols of uninterpreted sorts and uninterpreted functions

Axioms:

- Reflexivity: $\forall x.\ x = x$
- Symmetry: $\forall x, y.\ x = y \rightarrow y = x$
- Transitivity: $\forall x, y, z.\ x = y \wedge y = z \rightarrow x = z$
- **Congruence**:
  $\forall t_1, \ldots, t_n, t'_1, \ldots, t'_n. \bigwedge_{i=1}^{n} t_i = t'_i \rightarrow F(t_1, \ldots, t_n) = F(t'_1, \ldots, t'_n)$

Decision procedure for EUF - An example

$$\phi^{UF} := x_1 = x_2 \wedge x_2 = x_3 \wedge F(x_1) = F(x_3) \wedge \neg(F(F(x_1)) = F(F(x_2)))$$

Decision procedure for EUF - An example

$$\phi^{UF} := x_1 = x_2 \wedge x_2 = x_3 \wedge F(x_1) = F(x_3) \wedge \neg(F(F(x_1)) = F(F(x_2)))$$

- $\{\{x_1, x_2\}, \{x_2, x_3\}, \{F(x_1), F(x_3)\}, \{F(x_2)\}, \{F(F(x_1))\}, \{F(F(x_2))\}\}$

Decision procedure for EUF - An example

$$\phi^{UF} := x_1 = x_2 \wedge x_2 = x_3 \wedge F(x_1) = F(x_3) \wedge \neg(F(F(x_1)) = F(F(x_2)))$$

- $\{\{x_1, x_2\}, \{x_2, x_3\}, \{F(x_1), F(x_3)\}, \{F(x_2)\}, \{F(F(x_1))\}, \{F(F(x_2))\}\}$
- $\{\{x_1, x_2, x_3\}, \{F(x_1), F(x_3)\}, \{F(x_2)\}, \{F(F(x_1))\}, \{F(F(x_2))\}\}$

Decision procedure for EUF - An example

$$\phi^{UF} := x_1 = x_2 \land x_2 = x_3 \land F(x_1) = F(x_3) \land \neg(F(F(x_1)) = F(F(x_2)))$$

- $\{\{x_1, x_2\}, \{x_2, x_3\}, \{F(x_1), F(x_3)\}, \{F(x_2)\}, \{F(F(x_1))\}, \{F(F(x_2))\}\}$
- $\{\{x_1, x_2, x_3\}, \{F(x_1), F(x_3)\}, \{F(x_2)\}, \{F(F(x_1))\}, \{F(F(x_2))\}\}$
- $\{\{x_1, x_2, x_3\}, \{F(x_1), F(x_2), F(x_3)\}, \{F(F(x_1))\}, \{F(F(x_2))\}\}$

Decision procedure for EUF - An example

$$\phi^{UF} := x_1 = x_2 \wedge x_2 = x_3 \wedge F(x_1) = F(x_3) \wedge \neg(F(F(x_1)) = F(F(x_2)))$$

- $\{\{x_1, x_2\}, \{x_2, x_3\}, \{F(x_1), F(x_3)\}, \{F(x_2)\}, \{F(F(x_1))\}, \{F(F(x_2))\}\}$
- $\{\{x_1, x_2, x_3\}, \{F(x_1), F(x_3)\}, \{F(x_2)\}, \{F(F(x_1))\}, \{F(F(x_2))\}\}$
- $\{\{x_1, x_2, x_3\}, \{F(x_1), F(x_2), F(x_3)\}, \{F(F(x_1))\}, \{F(F(x_2))\}\}$
- $\{\{x_1, x_2, x_3\}, \{F(x_1), F(x_2), F(x_3)\}, \{F(F(x_1)), F(F(x_2))\}\}$

Decision procedure for EUF - An example

$$\phi^{UF} := x_1 = x_2 \wedge x_2 = x_3 \wedge F(x_1) = F(x_3) \wedge \boxed{\neg(F(F(x_1)) = F(F(x_2))}\,)$$

- $\{\{x_1, x_2\}, \{x_2, x_3\}, \{F(x_1), F(x_3)\}, \{F(x_2)\}, \{F(F(x_1))\}, \{F(F(x_2))\}\}$
- $\{\{x_1, x_2, x_3\}, \{F(x_1), F(x_3)\}, \{F(x_2)\}, \{F(F(x_1))\}, \{F(F(x_2))\}\}$
- $\{\{x_1, x_2, x_3\}, \{F(x_1), F(x_2), F(x_3)\}, \{F(F(x_1))\}, \{F(F(x_2))\}\}$
- $\{\{x_1, x_2, x_3\}, \{F(x_1), F(x_2), F(x_3)\}, \boxed{\{F(F(x_1)), F(F(x_2))\}}\}$
- Contradict! Return **UNSAT**.

Decision procedure for EUF - Congruence Closure

Input: $\phi^{UF} :=$ conjunction of equality literals

1. Compute the **congruence closure**
   a. Put two terms t_1, t_2 in an equivalence class if $t_1 = t_2$ is in ϕ^{UF}.
   b. Merge two equivalence classes C_1, C_2 if
      $\exists t.\ t \in C_1 \land t \in C_2$.
   c. Merge two equivalence classes C_1, C_2 if
      $\exists t_1, t_2, C_3.\ t_1 \in C_3 \land t_2 \in C_3 \land F(t_1) \in C_1 \land F(t_2) \in C_2$.
2. For every literal $\neg(t_i = t_j)$ in ϕ^{UF}, if t_i, t_j are in the same equivalence class, return **UNSAT**. Otherwise, return **SAT**.

Decision procedure for EUF - Exercise

1. Apply the decision procedure for EUF, how many equivalence classes are left after execution?
   - $\phi^{UF} := x_1 = x_2 \wedge F^4(x_2) = F^5(x_3) \wedge F(x_3) = x_1$
   - $\phi^{UF} := F(x_1) = x_2 \wedge \neg(F^3(x_1) = F^4(x_3)) \wedge F^3(x_3) = F(x_2)$
2. Apply DPLL(EUF) (including the decision procedure for EUF):
   - $\phi := (\neg(F(b) = c) \ \vee \ F(a) = F^2(b)) \ \wedge \ a = c$
     $\wedge \ (\neg(F^4(b) = F^3(c)) \ \vee \ F(b) = c)$

Application - Prove Program Equivalence

- Scheme: Two programs a, b with bounded loops
- Basic idea:
  1. Unroll the loops, and for each assignment instruction, replace the left-hand side with an auxiliary variable and then join them.
  2. Replace each interpreted function with an uninterpreted function in order to acquire ϕ_a^{UF} and ϕ_b^{UF}.
  3. Prove program equivalence by solving:

$$input_a = input_b \land \phi_a^{UF} \land \phi_b^{UF} \rightarrow output_a = output_b$$

```
int power3(int in)
{
  int i, out_a;
  out_a = in;
  for (i = 0; i < 2; i++)
    out_a = out_a * in;
  return out_a;
}
```

```
int power3_new(int in)
{
  int out_b;

  out_b = (in * in) * in;

  return out_b;
}
```

Application - Prove Program Equivalence

Unroll the loop in power3(),

$$\begin{aligned} out0\_a &= in0_a && \wedge \\ out1\_a &= out0\_a * in0_a && \wedge \\ out2\_a &= out1\_a * in0_a \end{aligned} \qquad \Bigg| \qquad out0\_b = (in0_b * in0_b) * in0_b;$$

Replace ' * ' with uninterpreted function ' G ',

$$\begin{aligned} out0\_a &= in0\_a && \wedge \\ out1\_a &= G(out0\_a, in0\_a) && \wedge \\ out2\_a &= G(out1\_a, in0\_a) \end{aligned} \qquad out0\_b = G(G(in0\_b, in0\_b), in0\_b)$$

Then we obtain:

$$in0\_a = in0\_b \wedge \phi_a^{UF} \wedge \phi_b^{UF} \rightarrow out2\_a = out2\_b$$

Table of Contents

Theory of Arrays

Signature:

$$\Sigma := \{=, \cdot[\cdot], \cdot\{\cdot \leftarrow \cdot\}\}$$

- $a[i]$ (**Read**) represents the value of array a at index i
- $a\{i \leftarrow x\}$ (**Write**) represents the copy of array a with the value at index i replaced by x

Axioms:

- Reflexivity, Symmetry and Transitivity axioms from Equality theory
- **Array Congruence**: $\forall a_1, a_2, i, j.\ a_1 = a_2 \wedge i = j \rightarrow a_1[i] = a_2[j]$
- **Read-Over-Write**: $\forall a, x, i, j.\ a\{i \leftarrow x\}[j] = \begin{cases} x & \text{for} \quad i = j \\ a[j] & \text{for} \quad \neg(i = j) \end{cases}$

$$\phi^A := a\{i_1 \leftarrow x\}[j_1] = u \wedge a\{i_2 \leftarrow y\}[j_2] = v \wedge \neg(i_1 = j_1) \wedge \neg(y = v)$$

Decision procedure for Arrays - An Example

$$\phi^A := \boxed{a\{i_1 \leftarrow x\}[j_1] = u} \wedge \boxed{a\{i_2 \leftarrow y\}[j_2] = v} \wedge \neg(i_1 = j_1) \wedge \neg(y = v)$$

$$\phi^A_{rewrite} := \boxed{((i_1 = j_1 \wedge x = u) \vee (\neg(i_1 = j_1) \wedge a[j_1] = u))} \wedge$$
$$\boxed{((i_2 = j_2 \wedge y = v) \vee (\neg(i_2 = j_2) \wedge a[j_2] = v))} \wedge \neg(i_1 = j_1) \wedge \neg(y = v)$$

Decision procedure for Arrays - An Example

$$\phi^A := \ a\{i_1 \leftarrow x\}[j_1] = u \ \wedge \ a\{i_2 \leftarrow y\}[j_2] = v \ \wedge \neg(i_1 = j_1) \wedge \neg(y = v)$$

$$\begin{aligned}\phi^A_{rewrite} :=& ((i_1 = j_1 \wedge x = u) \vee (\neg(i_1 = j_1) \wedge \boxed{a[j_1]} = u)) \wedge \\ & ((i_2 = j_2 \wedge y = v) \vee (\neg(i_2 = j_2) \wedge \boxed{a[j_2]} = v)) \wedge \neg(i_1 = j_1) \wedge \neg(y = v)\end{aligned}$$

$$\begin{aligned}\phi' :=& ((i_1 = j_1 \wedge x = u) \vee (\neg(i_1 = j_1) \wedge \boxed{F_a(j_1)} = u)) \wedge \\ & ((i_2 = j_2 \wedge y = v) \vee (\neg(i_2 = j_2) \wedge \boxed{F_a(j_2)} = v)) \wedge \neg(i_1 = j_1) \wedge \neg(y = v)\end{aligned}$$

Decision procedure for Arrays - An Example

$$\phi' := ((i_1 = j_1 \land x = u) \lor (\neg(i_1 = j_1) \land F_a(j_1) = u)) \land$$
$$((i_2 = j_2 \land y = v) \lor (\neg(i_2 = j_2) \land F_a(j_2) = v)) \land \neg(i_1 = j_1) \land \neg(y = v)$$

$$\phi'_p := ((a_0 \land a_1) \lor (\neg a_0 \land a_2)) \land ((a_3 \land a_4) \lor (\neg a_3 \land a_5)) \land \neg a_0 \land \neg a_4$$

Decision procedure for Arrays - An Example

$$\phi' := ((i_1 = j_1 \wedge x = u) \vee (\neg(i_1 = j_1) \wedge F_a(j_1) = u)) \wedge$$
$$((i_2 = j_2 \wedge y = v) \vee (\neg(i_2 = j_2) \wedge F_a(j_2) = v)) \wedge \neg(i_1 = j_1) \wedge \neg(y = v)$$

$$\phi'_p := ((a_0 \wedge a_1) \vee (\neg a_0 \wedge a_2)) \wedge ((a_3 \wedge a_4) \vee (\neg a_3 \wedge a_5)) \wedge \neg a_0 \wedge \neg a_4$$

- Propagate: $a_0 \mapsto \mathsf{F}, a_4 \mapsto \mathsf{F}$

Decision procedure for Arrays - An Example

$$\begin{aligned}\phi' :=& ((i_1 = j_1 \land x = u) \lor (\neg(i_1 = j_1) \land F_a(j_1) = u)) \land \\ & ((i_2 = j_2 \land y = v) \lor (\neg(i_2 = j_2) \land F_a(j_2) = v)) \land \neg(i_1 = j_1) \land \neg(y = v)\end{aligned}$$

$$\phi'_p := ((a_0 \land a_1) \lor (\neg a_0 \land a_2)) \land ((a_3 \land a_4) \lor (\neg a_3 \land a_5)) \land \neg a_0 \land \neg a_4$$

- Propagate: $a_0 \mapsto \mathsf{F}, a_4 \mapsto \mathsf{F}$
- Decide: $a_3 \mapsto \mathsf{T}$

Decision procedure for Arrays - An Example

$$\phi' := ((i_1 = j_1 \wedge x = u) \vee (\neg(i_1 = j_1) \wedge F_a(j_1) = u)) \wedge$$
$$((i_2 = j_2 \wedge y = v) \vee (\neg(i_2 = j_2) \wedge F_a(j_2) = v)) \wedge \neg(i_1 = j_1) \wedge \neg(y = v)$$

$$\phi'_p := ((a_0 \wedge a_1) \vee (\neg a_0 \wedge a_2)) \wedge ((a_3 \wedge a_4) \vee (\neg a_3 \wedge a_5)) \wedge \neg a_0 \wedge \neg a_4$$

- Propagate: $a_0 \mapsto \mathrm{F}, a_4 \mapsto \mathrm{F}$
- Decide: $a_3 \mapsto \mathrm{T}$
- Solve: $(\neg(i_1 = j_1) \wedge \neg(y = v) \wedge i_2 = j_2)$, **SAT**
  $\Rightarrow$ Return **SAT**

Decision procedure for Arrays

Input: $\phi^A :=$ conjunction of array literals

Basic idea:

1. According to Read-Over-Write axiom, we can branch a Write term $a\{i \leftarrow x\}[j]$ into two cases:
   - x for $i = j$
   - $a[j]$ for $\neg(i = j)$
2. Recursive on step 1 until ϕ^A contains only Read terms, then replace each term $a[i]$ with an uninterpreted function term $F_a(i)$ to obtain ϕ'.
3. The remaining part is same as solving an EUF formula.

Decision procedure for Arrays - Exercise

1. Apply the decision procedure for arrays:
   - $\phi^A := \neg(i = j) \wedge \neg(i = k) \wedge a\{j \leftarrow v\}[i] = a\{k \leftarrow w\}[j]$

2. Apply DPLL(Arrays) (including the decision procedure for arrays):
   - $\phi := (i = j \vee \neg(i = j)) \wedge (i = k \vee \neg(i = k)) \wedge a\{j \leftarrow v\}[i] = a\{k \leftarrow w\}[j]$

Table of Contents

Combined theories

- In the previous example, we only invoked one theory solver. But in practice, we often encounter a combination of theories.

- For example:

$$
\begin{aligned}
\phi :=& (\neg(u = G(v) \vee \neg(u = v)) \wedge \neg(a[u] = a[G(u)]) \\
& \wedge a[G(u)] = a\{G(v) \leftarrow x\}[u]
\end{aligned}
$$

- Purify(EUF/ARRAY):

$$
\begin{aligned}
\phi_{purified} :=& (\neg(u = G(v)) \vee \neg(u = v)) \wedge \neg(a[u] = a[y_1]) \\
& \wedge a[y_1] = a\{y_2 \leftarrow x\}[u] \wedge y_1 = G(u) \wedge y_2 = G(v)
\end{aligned}
$$

Combined theories - An Example

$$\phi_{purified} := (\neg(u = G(v)) \vee \neg(u = v)) \wedge \neg(a[u] = a[y_1])$$
$$\wedge\ a[y_1] = a\{y_2 \leftarrow x\}[u] \wedge y_1 = G(u) \wedge y_2 = G(v)$$

$$\phi_p := (\neg a_0 \vee \neg a_1) \wedge \neg a_2 \wedge a_3 \wedge a_4 \wedge a_5$$

- Propagate: $a_2 \mapsto \mathrm{F}, a_3 \mapsto \mathrm{T}, a_4 \mapsto \mathrm{T}, a_5 \mapsto \mathrm{T}$

Combined theories - An Example

$$\phi_{purified} := (\neg(u = G(v)) \vee \neg(u = v)) \wedge \neg(a[u] = a[y_1])$$
$$\wedge\ a[y_1] = a\{y_2 \leftarrow x\}[u] \wedge y_1 = G(u) \wedge y_2 = G(v)$$

$$\phi_p := (\neg a_0 \vee \neg a_1) \wedge \neg a_2 \wedge a_3 \wedge a_4 \wedge a_5$$

- Propagate: $a_2 \mapsto \mathsf{F}, a_3 \mapsto \mathsf{T}, a_4 \mapsto \mathsf{T}, a_5 \mapsto \mathsf{T}$
- Decide: $a_0 \mapsto \mathsf{F}$

Combined theories - An Example

$$\phi_{purified} := (\neg(u = G(v)) \vee \neg(u = v)) \wedge \neg(a[u] = a[y_1])$$
$$\wedge\ a[y_1] = a\{y_2 \leftarrow x\}[u] \wedge y_1 = G(u) \wedge y_2 = G(v)$$

$$\phi_p := (\neg a_0 \vee \neg a_1) \wedge \neg a_2 \wedge a_3 \wedge a_4 \wedge a_5$$

- Propagate: $a_2 \mapsto \mathsf{F}, a_3 \mapsto \mathsf{T}, a_4 \mapsto \mathsf{T}, a_5 \mapsto \mathsf{T}$
- Decide: $a_0 \mapsto \mathsf{F}$
- Pass assignment $\alpha_1 := \{a_0 \mapsto \mathsf{F}, a_3 \mapsto \mathsf{T}, a_4 \mapsto \mathsf{T}\}$ to EUF solver, and assignment $\alpha_2 := \{a_2 \mapsto \mathsf{F}, a_3 \mapsto \mathsf{T}\}$ to ARRAY solver

Combined theories - An Example

$$\phi_{purified} := (\neg(u = G(v)) \vee \neg(u = v)) \wedge \neg(a[u] = a[y_1])$$
$$\wedge\ a[y_1] = a\{y_2 \leftarrow x\}[u] \wedge y_1 = G(u) \wedge y_2 = G(v)$$
$$\phi_p := (\neg a_0 \vee \neg a_1) \wedge \neg a_2 \wedge a_3 \wedge a_4 \wedge a_5$$

- Propagate: $a_2 \mapsto \mathrm{F}, a_3 \mapsto \mathrm{T}, a_4 \mapsto \mathrm{T}, a_5 \mapsto \mathrm{T}$
- Decide: $a_0 \mapsto \mathrm{F}$
- Pass assignment $\alpha_1 := \{a_0 \mapsto \mathrm{F}, a_3 \mapsto \mathrm{T}, a_4 \mapsto \mathrm{T}\}$ to EUF solver, assignment $\alpha_2 := \{a_2 \mapsto \mathrm{F}, a_3 \mapsto \mathrm{T}\}$ to ARRAY solver,
- EUF: solves $(\neg(u = G(v)) \wedge y_1 = G(u) \wedge y2 = G(v))$, gets **SAT**, ARRAY: solves $(\neg(a[u] = a[y_1]) \wedge a[y_1] = a\{y_2 \leftarrow x\}[u])$, gets **SAT**.

Combined theories - An Example

- Both theory solvers got **SAT**, but can we conclude that ϕ is **SAT**?

Combined theories - An Example

- Both theory solvers get **SAT**, but can we conclude that ϕ is **SAT**?

  Not yet, theory solvers must agree on shared variables!
  (u, y_1, y_2 in this case)

Combined theories - An Example

- Both theory solvers get **SAT**, but can we conclude that ϕ is **SAT**?

  Not yet, theory solvers must agree on shared variables!
  (u, y_1, y_2 in this case)

- For EUF, $(\neg(u = G(v)) \wedge y_1 = G(u) \wedge y2 = G(v))$ implies $\neg(u = y_2)$.
  For ARRAY, $(\neg(a[u] = a[y_1]) \wedge a[y_1] = a\{y_2 \leftarrow x\}[u])$ implies $u = y_2$.

- Both theory solvers get **SAT**, but can we conclude that ϕ is **SAT**?

  Not yet, theory solvers must agree on shared variables!
  (u, y_1, y_2 in this case)

- For EUF, $(\neg(u = G(v)) \wedge y_1 = G(u) \wedge y2 = G(v))$ implies $\neg(u = y_2)$.
  For ARRAY, $(\neg(a[u] = a[y_1]) \wedge a[y_1] = a\{y_2 \leftarrow x\}[u])$ implies $u = y_2$.

  $\Rightarrow$The solvers do not agree on share variables.

Combined theories - An Example

$$\phi_{purified} := (\neg(u = G(v)) \vee \neg(u = v)) \wedge \neg(a[u] = a[y_1])$$
$$\wedge\ a[y_1] = a\{y_2 \leftarrow x\}[u] \wedge y_1 = G(u) \wedge y_2 = G(v)$$
$$\wedge\ (u = y_2 \vee \neg(u = y_2))$$

$$\phi_p := (\neg a_0 \vee \neg a_1) \wedge \neg a_2 \wedge a_3 \wedge a_4 \wedge a_5 \wedge (a_6 \vee \neg a_6)$$

- Propagate: $a_2 \mapsto \mathrm{F}, a_3 \mapsto \mathrm{T}, a_4 \mapsto \mathrm{T}, a_5 \mapsto \mathrm{T}$
- Decide: $a_0 \mapsto \mathrm{F}$
- Pass assignment $\alpha_1 := \{a_0 \mapsto \mathrm{F}, a_4 \mapsto \mathrm{T}, a_5 \mapsto \mathrm{T}\}$ to EUF solver, assignment $\alpha_2 := \{a_2 \mapsto \mathrm{F}, a_3 \mapsto \mathrm{T}\}$ to ARRAY solver,
- EUF: solves $(\neg(u = G(v)) \wedge y_1 = G(u) \wedge y2 = G(v))$ and get **SAT**, ARRAY: solves $(\neg(a[u] = a[y_1]) \wedge a[y_1] = a\{y_2 \leftarrow x\}[u])$ and get **SAT**. The solvers do not agree on share variables, add blocking clauses.

Combined theories - An Example

$$\phi_{purified} := (\boxed{\neg(u = G(v))} \vee \neg(u = v)) \wedge \boxed{\neg(a[u] = a[y_1])}$$
$$\wedge \boxed{a[y_1] = a\{y_2 \leftarrow x\}[u]} \wedge \boxed{y_1 = G(u)} \wedge \boxed{y_2 = G(v)}$$
$$\wedge \boxed{(u = y_2 \vee \neg(u = y_2))}$$

$$\phi_p := (\boxed{\neg a_0} \vee \neg a_1) \wedge \boxed{\neg a_2} \wedge \boxed{a_3} \wedge \boxed{a_4} \wedge \boxed{a_5} \wedge \boxed{(a_6 \vee \neg a_6)}$$

- Propagate: $a_2 \mapsto \mathrm{F}, a_3 \mapsto \mathrm{T}, a_4 \mapsto \mathrm{T}, a_5 \mapsto \mathrm{T}$
- Decide: $a_0 \mapsto \mathrm{F}$
- Pass assignment $\alpha_1 := \{a_0 \mapsto \mathrm{F}, a_4 \mapsto \mathrm{T}, a_5 \mapsto \mathrm{T}\}$ to EUF solver, assignment $\alpha_2 := \{a_2 \mapsto \mathrm{F}, a_3 \mapsto \mathrm{T}\}$ to ARRAY solver,
- EUF: solves $(\neg(u = G(v)) \wedge y_1 = G(u) \wedge y2 = G(v))$ and get **SAT**, ARRAY: solves $(\neg(a[u] = a[y_1]) \wedge a[y_1] = a\{y_2 \leftarrow x\}[u])$ and get **SAT**. The solvers does not agree on share variables, add blocking clauses.
- Conflict! Backtrack to the decision.

Combined theories - An Example

$$\phi_{purified} := (\neg(u = G(v)) \lor \boxed{\neg(u = v)}) \land \boxed{\neg(a[u] = a[y_1])}$$
$$\land \boxed{a[y_1] = a\{y_2 \leftarrow x\}[u]} \land \boxed{y_1 = G(u)} \land \boxed{y_2 = G(v)}$$
$$\land (u = y_2 \lor \neg(u = y_2))$$

$$\phi_p := (\neg a_0 \lor \boxed{\neg a_1}) \land \boxed{\neg a_2} \land \boxed{a_3} \land \boxed{a_4} \land \boxed{a_5} \land (a_6 \lor \neg a_6)$$

- Propagate: $a_2 \mapsto \mathsf{F}, a_3 \mapsto \mathsf{T}, a_4 \mapsto \mathsf{T}, a_5 \mapsto \mathsf{T}$
- Backtrack: $a_1 \mapsto \mathsf{F}$

Combined theories - An Example

$$\phi_{purified} := (\neg(u = G(v)) \vee \neg(u = v)) \wedge \neg(a[u] = a[y_1])$$
$$\wedge \; a[y_1] = a\{y_2 \leftarrow x\}[u] \wedge y_1 = G(u) \wedge y_2 = G(v)$$
$$\wedge \, (u = y_2 \vee \neg(u = y_2))$$

$$\phi_p := (\neg a_0 \vee \neg a_1) \wedge \neg a_2 \wedge a_3 \wedge a_4 \wedge a_5 \wedge (a_6 \vee \neg a_6)$$

- Propagate: $a_2 \mapsto \mathsf{F}, a_3 \mapsto \mathsf{T}, a_4 \mapsto \mathsf{T}, a_5 \mapsto \mathsf{T}$
- Backtrack: $a_1 \mapsto \mathsf{F}$
- Decide: $a_6 \mapsto \mathsf{T}$

Combined theories - An Example

$$\begin{aligned}\phi_{purified} :=(& \neg(u = G(v)) \vee \neg(u = v)) \wedge \neg(a[u] = a[y_1]) \\ & \wedge a[y_1] = a\{y_2 \leftarrow x\}[u] \wedge y_1 = G(u) \wedge y_2 = G(v) \\ & \wedge (u = y_2 \vee \neg(u = y_2))\end{aligned}$$

$$\phi_p := (\neg a_0 \vee \neg a_1) \wedge \neg a_2 \wedge a_3 \wedge a_4 \wedge a_5 \wedge (a_6 \vee \neg a_6)$$

- Propagate: $a_2 \mapsto \mathrm{F}, a_3 \mapsto \mathrm{T}, a_4 \mapsto \mathrm{T}, a_5 \mapsto \mathrm{T}$
- Backtrack: $a_1 \mapsto \mathrm{F}$
- Decide: $a_6 \mapsto \mathrm{T}$
- Pass assignment $\alpha_1 := \{a_1 \mapsto \mathrm{F}, a_4 \mapsto \mathrm{T}, a_5 \mapsto \mathrm{T}, a_6 \mapsto \mathrm{T}\}$ to EUF solver, assignment $\alpha_2 := \{a_2 \mapsto \mathrm{F}, a_3 \mapsto \mathrm{T}\}$ to ARRAY solver,

Combined theories - An Example

$$\phi_{purified} := (\neg(u = G(v)) \vee \boxed{\neg(u = v)}) \wedge \boxed{\neg(a[u] = a[y_1])}$$
$$\wedge \boxed{a[y_1] = a\{y_2 \leftarrow x\}[u]} \wedge \boxed{y_1 = G(u)} \wedge \boxed{y_2 = G(v)}$$
$$\wedge (\boxed{u = y_2} \vee \boxed{\neg(u = y_2)})$$
$$\phi_p := (\neg a_0 \vee \boxed{\neg a_1}) \wedge \boxed{\neg a_2} \wedge \boxed{a_3} \wedge \boxed{a_4} \wedge \boxed{a_5} \wedge (\boxed{a_6} \vee \boxed{\neg a_6})$$

- Propagate: $a_2 \mapsto \mathrm{F}, a_3 \mapsto \mathrm{T}, a_4 \mapsto \mathrm{T}, a_5 \mapsto \mathrm{T}$
- Backtrack: $a_1 \mapsto \mathrm{F}$
- Decide: $a_6 \mapsto \mathrm{T}$
- Pass assignment $\alpha_1 := \{a_1 \mapsto \mathrm{F}, a_4 \mapsto \mathrm{T}, a_5 \mapsto \mathrm{T}, a_6 \mapsto \mathrm{T}\}$ to EUF solver, assignment $\alpha_2 := \{a_2 \mapsto \mathrm{F}, a_3 \mapsto \mathrm{T}\}$ to ARRAY solver,
- EUF: solves $(\neg(u = v) \wedge y_1 = G(u) \wedge y2 = G(v) \wedge u = G(v))$, gets **SAT**,
  ARRAY: solves $(\neg(a[u] = a[y_1]) \wedge a[y_1] = a\{y_2 \leftarrow x\}[u])$,gets **SAT**

Combined theories - An Example

$$\phi_{purified} := (\neg(u = G(v)) \lor \neg(u = v)) \land \neg(a[u] = a[y_1])$$
$$\land\ a[y_1] = a\{y_2 \leftarrow x\}[u] \land y_1 = G(u) \land y_2 = G(v)$$
$$\land\ (u = y_2 \lor \neg(u = y_2))$$

$$\phi_p := (\neg a_0 \lor \neg a_1) \land \neg a_2 \land a_3 \land a_4 \land a_5 \land (a_6 \lor \neg a_6)$$

- Propagate: $a_2 \mapsto \mathsf{F}, a_3 \mapsto \mathsf{T}, a_4 \mapsto \mathsf{T}, a_5 \mapsto \mathsf{T}$
- Backtrack: $a_1 \mapsto \mathsf{F}$
- Decide: $a_6 \mapsto \mathsf{T}$
- Pass assignment $\alpha_1 := \{a_1 \mapsto \mathsf{F}, a_4 \mapsto \mathsf{T}, a_5 \mapsto \mathsf{T}, a_6 \mapsto \mathsf{T}\}$ to EUF solver, assignment $\alpha_2 := \{a_2 \mapsto \mathsf{F}, a_3 \mapsto \mathsf{T}\}$ to ARRAY solver,

Combined theories - An Example

- EUF: solves $(\neg(u = v) \wedge y_1 = G(u) \wedge y2 = G(v) \wedge u = G(v))$, gets **SAT**,
  ARRAY: solves $(\neg(a[u] = a[y_1]) \wedge a[y_1] = a\{y_2 \leftarrow x\}[u])$, gets **SAT**
- Both solvers agree on share variables, return **SAT**.

Implementation for Combined theories

- Main ideas:
  1. Purify the literals(each literal contains one theory).
  2. Once the SAT solver finds an assignment, pass the corresponding part to each theory solver.
  3. If any of the solvers gets **UNSAT**, then return **UNSAT**.
  4. If every theory solver gets **SAT**, then check if they agree on shared variables. If so, return **SAT**, otherwise, backtrack and go to step 2.
- The decision procedure above requires the theories to satisfy several properties:
  1. First-order, quantifier-free, decidable theories with equality.
  2. Have disjoint signatures, except "=".
  3. Interpreted over an infinite domain.

Combined theories - Exercise

Apply the decision procedure for combined theories to solve the formula: (you may omit the decision procedure for each theory solver, but must include the discussion of whether they agree on shared variables)

- $\phi := G(F(x_1 - 2)) = x_1 + 2 \land G(F(x_2)) = x_2 - 2 \land (x_2 + 1 = x_1 - 1)$

Key Takeaways

- The basic architecture and algorithm of an SMT solver
- The elementary decision procedure for the theory of EUF and Arrays
- The implementation of an SMT solver for combined theories

Reference

[1] Nikolaj Bjørner. "Satisfiability Modulo Theories". https://www.microsoft.com/en-us/research/wp-content/uploads/2017/02/MOD-12-8-2015.pptx. 2015.

[2] Ofer Strichman Daniel Kroening. *Decision Procedures, An Algorithmic Point of View (2nd ed.)* Springer, 2016.

[3] Andrew Reynolds. "DPLL(T) for SMT". https://resources.mpi-inf.mpg.de/departments/rg1/conferences/vtsa17/slides/reynolds-vtsa-part1.pdf. 2017.

[4] Ming-Hsien Tsai. "Software Verification with Satisfiability Modulo Theories - Decision Procedures". https://flolac.iis.sinica.edu.tw/flolac17/lib/exe/fetch.php%3Fmedia=course:02_euf.pdf. 2017.

[5] Ming-Hsien Tsai. "Software Verification with Satisfiability Modulo Theories - Introduction". https://flolac.iis.sinica.edu.tw/flolac17/lib/exe/fetch.php%3Fmedia=course:01_smt.pdf. 2017.