

— SMT Workshop, Princeton 2008 —

**Deciding Array Formulas
with Frugal Axiom Instantiation**

Amit Goel and Sava Krstić

Intel Strategic CAD Labs

Signature: $\{\text{read}, \text{write}\}$

Read-over-Write: $\text{read}(\text{write}(a, i, x), j) = \begin{cases} x & \text{if } i = j \\ \text{read}(a, j) & \text{otherwise} \end{cases}$

Extensionality: $a \neq b \Rightarrow \exists i. \text{read}(a, i) \neq \text{read}(b, i)$

- First decision procedure for satisfiability of sets of literals modulo this theory: [Stump Dill Barrett Levitt, 2001]

Example*

(RoW Axiom) $\forall a b i x. b = \text{write}(a, i, x) \wedge i \neq j \rightarrow \text{read}(b, j) = \text{read}(a, j)$

$b = \text{write}(a, i, x)$ $j \neq k$ $\text{read}(b, j) \neq \text{read}(a, j)$ $\text{read}(b, k) \neq \text{read}(a, k)$	$b = \text{write}(a, i, x)$ $j \neq k$ $\text{read}(b, j) \neq \text{read}(a, j)$ $\text{read}(b, k) \neq \text{read}(a, k)$ $b = \text{write}(a, i, x) \wedge i \neq j \rightarrow \text{read}(b, j) = \text{read}(a, j)$ $b = \text{write}(a, i, x) \wedge i \neq k \rightarrow \text{read}(b, k) = \text{read}(a, k)$
Φ_1	Φ_2

- Φ_1 is unsatisfiable modulo the theory of arrays
- Φ_2 is unsatisfiable modulo the theory of equality (uninterpreted functions)

* ... for non-existence of quantifier-free interpolants for arrays (R. Majumdar)

Reduction to Uninterpreted Functions

Fact. Given a set of literals Φ , there exists a finite set A of axiom instances such that

Φ is satisfiable modulo theory of arrays

if and only if

$A \cup \Phi$ is satisfiable modulo EUF

- This and other reduction results are in [Kapur, Zarba · 2005]
- Problem: generate **small, but big enough** sets of instances.
- *Yices, Z3* do it. How exactly? Are they complete?

Notation Change

$$\text{read}(a, i) \equiv ai$$

$$\text{write}(a, i, x) \equiv U(a, i, x)$$

(RoW Axiom) $\forall abix. b = U(a, i, x) \wedge i \neq j \rightarrow bj = aj$

$b = U(a, i, x)$ $j \neq k$ $bj \neq aj$ $bk \neq ak$	$b = U(a, i, x)$ $j \neq k$ $bj \neq aj$ $bk \neq ak$ $b = U(a, i, x) \wedge i \neq j \rightarrow bj = aj$ $b = U(a, i, x) \wedge i \neq k \rightarrow bk = ak$
Φ_1	Φ_2

- Φ_1 is unsatisfiable modulo the theory of **updatable functions** (“arrays”)
- Φ_2 is unsatisfiable modulo the theory of equality (uninterpreted functions)

Our Contribution

Extend the congruence closure solver in a “DPLL(\mathcal{T})-with-splitting-on-demand” framework so that it is a complete solver for the theory \mathcal{U} of updatable functions.

- Define the meaning of “sound and complete” for theory solvers in this framework
- Define a solver for \mathcal{U} with a **frugal** axiom instantiation scheme
- Prove the solver is sound and **complete**
- Implementation in *DPT*

Purification Lemma

Every set of formulas is strongly equisatisfiable with a set of **smith**thereens:

(A) propositional clause

(B) $p \leftrightarrow P(x_1, \dots, x_m)$

(C) $z = f(y_1, \dots, y_n)$

where

- no p occurs on rhs in (B), (C)
 - **defset** property: there are no dependency cycles (p depends on x_i ; z depends on y_j)
- ✳ At initialization, the SAT solver gets (A), and each theory solver gets its part of (B)+(C).

Example

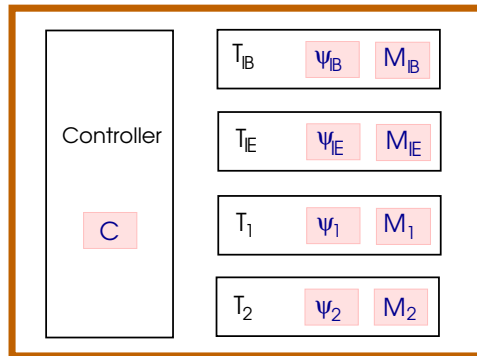
Input query:

$$U((\dots U(a_0, i_1, x_1), \dots), i_n, x_n) \neq U((\dots U(b_0, i_1, x_1), \dots), i_n, x_n)$$

After purification:

$$\begin{array}{llll} a_1 \stackrel{\Delta}{=} U(a_0, i_1, x_1) & a_2 \stackrel{\Delta}{=} U(a_1, i_2, x_2) & \dots & a_n \stackrel{\Delta}{=} U(a_{n-1}, i_n, x_n) \\ b_1 \stackrel{\Delta}{=} U(b_0, i_1, x_1) & b_2 \stackrel{\Delta}{=} U(b_1, i_2, x_2) & \dots & b_n \stackrel{\Delta}{=} U(b_{n-1}, i_n, x_n) \\ & p \stackrel{\Delta}{\leftrightarrow} (a_n = b_n) & & \neg p \end{array}$$

SMT Solvers



- $T_{\mathbb{B}}, T_{\mathbb{E}}$ are the SAT and EUF solvers
- $\Psi_{\mathbb{B}}$ is a set of clauses
- Ψ_i is a *defset* $\{x \stackrel{\Delta}{=} t, \dots p \stackrel{\Delta}{\leftrightarrow} \phi, \dots\}$
- M_i is the stack of asserted T_i -literals

Communication between modules

- $T_{\mathbb{B}} \longleftrightarrow T_i$ exchange of propositional literals
- $T_{\mathbb{E}} \longleftrightarrow T_i$ exchange of equalities b/w variables
- $T_i \longrightarrow T_{\mathbb{B}}$ theory lemmas as prop. clauses

Architecture derived from

- Abstract $DPLL(T)$ [Nieuwenhuis, Oliveras, Tinelli · 2005]
- “Splitting-on-demand” [Barrett, Nieuwenhuis, Oliveras, Tinelli · 2006]
- $NODPLL$ [Krstić, Goel · 2007]

Theory Solvers Abstractly

- Define a theory solver as a transition system
- Define solver's soundness and completeness
 - Makes an assumption on the solver's environment that lemmas obtained from the solver will not be contradicted by future literal assignments
- See paper for details

The Congruence Closure Solver

Literal	$\frac{l \in V_{\text{Bool}} \cup \neg V_{\text{Bool}} \quad l, \neg l \notin M}{M := M + l}$
Conflict	$\frac{\lceil u \neq v \rceil \in M \quad u \sim v \quad \text{status} = \text{no_cflct}}{\text{status} := \text{cflct}}$
Eq	$\frac{\lceil u = v \rceil \in M}{u \sim v}$
Congr	$\frac{i \sim j \quad a \sim b \quad \text{proxied}(ai) \quad \text{proxied}(bj)}{\lceil ai \rceil \sim \lceil bj \rceil}$

Notation:

- local state: \sim , an equivalence relation on V
- $\lceil u = v \rceil =_{\text{def}}$ the variable $p \in V_{\text{Bool}}$ such that $(p \overset{\Delta}{\leftrightarrow} u = v) \in \Psi$
- $\text{proxied}(ai) \Leftrightarrow_{\text{def}} \lceil ai \rceil$ is defined

The Congruence Closure Solver

Literal	$\frac{l \in V_{\text{Bool}} \cup \neg V_{\text{Bool}} \quad l, \neg l \notin M}{M := M + l}$
Conflict	$\frac{[u \neq v] \in M \quad u \sim v \quad \text{status} = \text{no_cflct}}{\text{status} := \text{cflct}}$
Eq	$\frac{[u = v] \in M}{u \sim v}$
Congr	$\frac{i \sim j \quad a \sim b \quad \text{proxied}(ai) \quad \text{proxied}(bj)}{[ai] \sim [bj]}$

Initialization:

- \sim_{init} has singleton classes
- Defset Ψ_{init} of the form $\{p \stackrel{\Delta}{\leftrightarrow} (u = v), \dots, x \stackrel{\Delta}{=} ai, \dots\}$

The Basic UPD Solver

$$\begin{array}{l}
 \text{RoW}_0 \quad \frac{a \times_i b \quad a, b, c \in V^{\sigma \Rightarrow \tau} \quad \text{proxied}(c_j)}{L := L + \{\text{proxy}(i \neq j \Rightarrow a_j = b_j)\}} \\
 \\
 \text{Ext}_0 \quad \frac{a, b \in V_{\text{init}}^{\sigma \Rightarrow \tau}}{L := L + \{\text{proxy}(a \neq b \Rightarrow a_i \neq b_i)\}} \quad (i \text{ is fresh})
 \end{array}$$

Notation:

- $a \times_i b$ means that $a = U(b, i, x)$ occurs in Ψ
- $V_{\text{init}}^{\sigma \Rightarrow \tau}$ is the initial set of variables of type $\sigma \Rightarrow \tau$.

Initialization:

- \sim_{init} has singleton classes
- Defset Ψ_{init} of the form $\{p \overset{\Delta}{\leftrightarrow} (u = v), \dots, x \overset{\Delta}{=} ai, \dots, a \overset{\Delta}{=} U(b, i, x), \dots\}$

A Frugal UPD Solver

RoW ₂	$\frac{a \times_i b \quad (c \sim a \text{ or } c \sim b) \quad \text{proxied}(cj) \quad i \not\sim j}{L := L + \{\text{proxy}(i \neq j \Rightarrow aj = bj)\}}$
Ext ₂	$\frac{a \bowtie b \quad \lceil a \neq b \rceil \in M}{L := L + \{\text{proxy}(a \neq b \Rightarrow ai \neq bi)\} \quad (i \text{ is fresh})}$
Ext_arg	$\frac{a \bowtie b \quad a \not\sim b \quad \text{proxied}(fa) \quad \text{proxied}(gb) \quad f \bowtie g}{\text{proxy}(a = b)}$

Notation:

- $a \times_i b$ means that $a = U(b, i, x)$ occurs in Ψ
- \bowtie is the equivalence relation generated by \sim and all \times_i

Initialization:

- Ensure $\text{proxied}(ai)$ if $a \stackrel{\Delta}{=} U(b, i, x)$ is in Ψ_{init}
- \sim_{init} generated by $\lceil ai \rceil \sim x$, where $a \stackrel{\Delta}{=} U(b, i, x)$ is in Ψ_{init}

Example

$$\begin{array}{llll} a_1 \stackrel{\Delta}{=} U(a_0, i_1, x_1) & a_2 \stackrel{\Delta}{=} U(a_1, i_2, x_2) & \dots & a_n \stackrel{\Delta}{=} U(a_{n-1}, i_n, x_n) \\ b_1 \stackrel{\Delta}{=} U(b_0, i_1, x_1) & b_2 \stackrel{\Delta}{=} U(b_1, i_2, x_2) & \dots & b_n \stackrel{\Delta}{=} U(b_{n-1}, i_n, x_n) \\ & p \stackrel{\Delta}{\leftrightarrow} (a_n = b_n) & & \neg p \end{array}$$

The update (\times) data:



Example: Counting Ext-lemmas



$$\text{Ext}_0 \quad \frac{a, b \in V_{\text{init}}^{\sigma \Rightarrow \tau}}{L := L + \{\text{proxy}(a \neq b \Rightarrow ai \neq bi)\} \quad (i \text{ is fresh})}$$

$\approx 2n^2$ lemmas
 n^2 "a-b" +
 $n^2/2$ "a-a" + $n^2/2$ "b-b"

$$\text{Ext}_1 \quad \frac{a \bowtie b \quad a \not\sim b}{L := L + \{\text{proxy}(a \neq b \Rightarrow ai \neq bi)\} \quad (i \text{ is fresh})}$$

$\approx n^2$ lemmas
 $n^2/2$ "a-a" + $n^2/2$ "b-b"

$$\text{Ext}_2 \quad \frac{a \bowtie b \quad [a \neq b] \in M}{L := L + \{\text{proxy}(a \neq b \Rightarrow ai \neq bi)\} \quad (i \text{ is fresh})}$$

$$\text{Ext_arg} \quad \frac{a \bowtie b \quad a \not\sim b \quad \text{proxied}(fa) \quad \text{proxied}(gb) \quad f \bowtie g}{\text{proxy}(a = b)}$$

0 lemmas

Example: Counting RoW-lemmas



$$\text{RoW}_0 \quad \frac{a \times_i b \quad a, b, c \in V^{\sigma \Rightarrow \tau} \quad \text{proxied}(c_j)}{L := L + \{\text{proxy}(i \neq j \Rightarrow a_j = b_j)\}}$$

$\approx 4n^3$ lemmas
($2n^2$ fresh vars)

$$\text{RoW}_1 \quad \frac{a \times_i b \quad c \bowtie a \quad \text{proxied}(c_j) \quad i \neq j}{L := L + \{\text{proxy}(i \neq j \Rightarrow a_j = b_j)\}}$$

$\approx n^3$ lemmas
($n^2/2$ fresh "a-a" vars)

$$\text{RoW}_2 \quad \frac{a \times_i b \quad c \sim a \text{ or } c \sim b \quad \text{proxied}(c_j) \quad i \neq j}{L := L + \{\text{proxy}(i \neq j \Rightarrow a_j = b_j)\}}$$

$\approx n^2$ lemmas
(not $2n^2$!)



Main Result

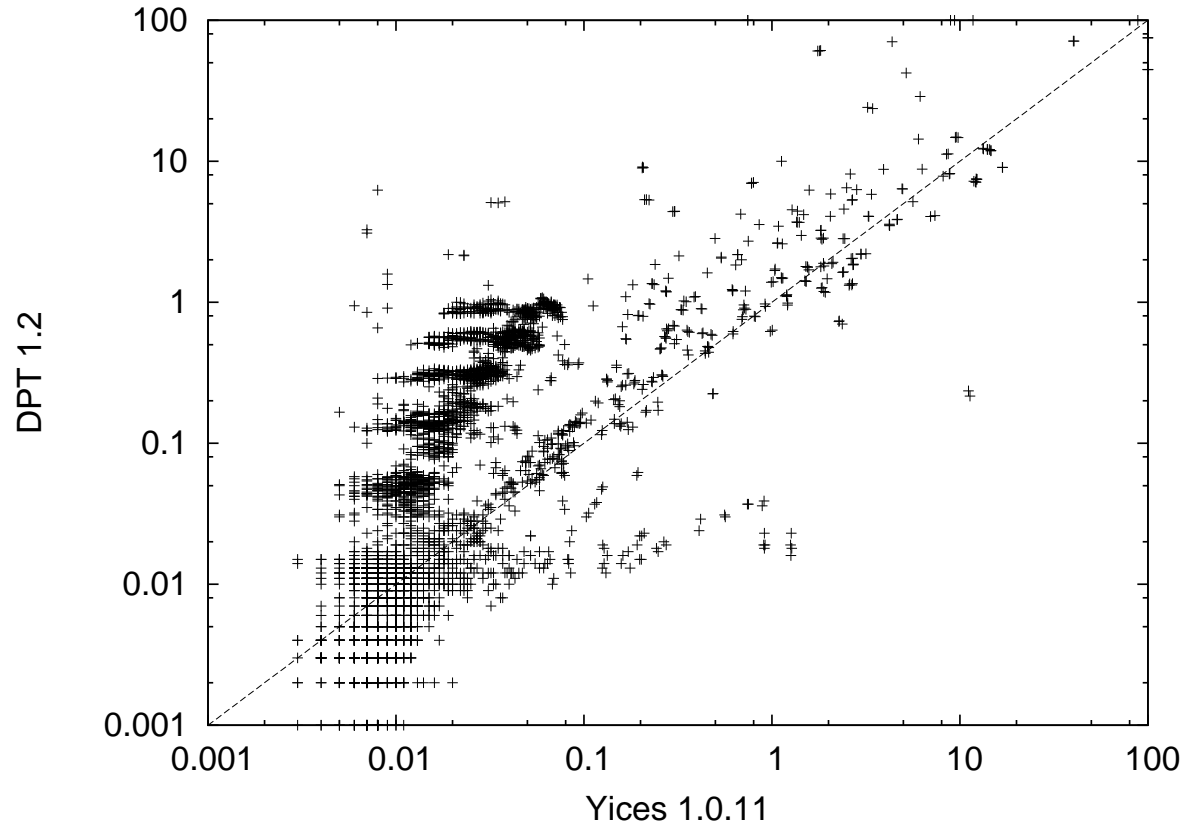
Theorem UPD is terminating, sound, and complete.

Implementation

DPT: Decision Procedure Toolkit

- SMT solver developed at Intel Strategic CAD Labs
- Goel, Grundy, Krstić, Fuchs
- written in OCaml
- open source (SourceForge)

Experimental Results



Full set of "QF_AUFIDL" SMT-LIB benchmarks

Other Array Solvers

- Stump, Dill, Barrett, Levitt (LICS'2001)
- Armando, Ranise, Rusinowitch (Inf. Comput., 2003)
- Bradley, Manna, Sipma (VMCI 2006)
- Ghilardi, Nicolini, Ranise, Zucchelli (Ann. Math. & AI, 2007)
- New *Barcelogic* solver (2008)
 - \boxtimes vs. \sim_I ; one could use both