A Write-Based Solver for SAT Modulo the Theory of Arrays

Miquel Bofill, Robert Nieuwenhuis,Albert Oliveras,
Enric Rodríguez-Carbonell and Albert Rubio

8th International Conference, FMCAD 2008
Portland, OR, USA
November 19th, 2008

UPC
Departament de Llenguatges i Sistemes Informàtics
UNIVERSITAT POLITÈCNICA DE CATALUNYA

Overview of the talk

- SAT Modulo Theories (SMT)
  - The Theory of Extensional Arrays
  - Solving SMT with DPLL(T)
- Handling Arrays in SMT
  - Theory instantiation for Arrays
  - A new solver for the theory of Arrays
- Key points
- Experimental evaluation
- Conclusions

Overview of the talk

- SAT Modulo Theories (SMT)
  - The Theory of Extensional Arrays
  - Solving SMT with DPLL(T)
- Handling Arrays in SMT
  - Theory instantiation for Arrays
  - A new solver for the theory of Arrays
- Key points
- Experimental evaluation
- Conclusions

Departament de Llenguatges i Sistemes Informatics
UNIVERSITAT POLITECNICA DE CATALUNYA

SAT Modulo Theories (SMT)

- Some problems are more naturally expressed in other logics than propositional logic, e.g:
  - Software verification needs reasoning about equality, arithmetic, data structures, ...
- SMT consists of deciding the satisfiability of a (ground) FO formula with respect to a background theory
- Example (Equality with Uninterpreted Functions – EUF):

$$g(a)=c \quad \wedge \quad (f(g(a)) \neq f(c) \vee g(a)=d) \quad \wedge \quad c \neq d$$

- Wide range of applications:
  - Predicate abstraction
  - Model checking
  - Equivalence checking
  - Static analysis
  - Scheduling
  - ...

UPC
Departament de Llenguatges i Sistemes Informàtics
UNIVERSITAT POLITÈCNICA DE CATALUNYA

The Theory of Extensional Arrays

- This is a very common structure

The Theory of Extensional Arrays

- This is a very common structure
- Axiomatization of the Theory:
  - Read/Write Axioms
    $i = j \Rightarrow read(write(a,i,x),j) = x$
    $i \neq j \Rightarrow read(write(a,i,x),j) = read(a,j)$
  - Extensionality
    $\forall i.read(a,i) = read(b,i) \Rightarrow a = b$

The Theory of Extensional Arrays

- This is a very common structure
- Axiomatization of the Theory:
  - Read/Write Axioms
    $i = j \Rightarrow read(write(a, i, x), j) = x$
    $i \neq j \Rightarrow read(write(a, i, x), j) = read(a, j)$
  - Extensionality
    $a \neq b \Rightarrow \exists i. read(a, i) \neq read(b, i)$

Departament de Llenguatges i Sistemes Informatics
UNIVERSITAT POLITECNICA DE CATALUNYA

The Theory of Extensional Arrays

- This is a very common structure
- Axiomatization of the Theory:
  - Read/Write Axioms
    $i = j \Rightarrow read(write(a, i, x), j) = x$
    $i \neq j \Rightarrow read(write(a, i, x), j) = read(a, j)$
  - Extensionality
    $a \neq b \Rightarrow \exists i. read(a, i) \neq read(b, i)$

Combined with
Uninterpreted Functions, Linear Integer Arithmetic or Bit-vectors

UPC
Departament de Llenguatges i Sistemes Informàtics
UNIVERSITAT POLITÈCNICA DE CATALUNYA

The Theory of Extensional Arrays

- This is a very common structure
- Axiomatization of the Theory:
  - Read/Write Axioms
    $i = j \Rightarrow read(write(a,i,x),j) = x$
    $i \neq j \Rightarrow read(write(a,i,x),j) = read(a,j)$
  - Extensionality
    $a \neq b \Rightarrow \exists i.read(a,i) \neq read(b,i)$

Combined with
Uninterpreted Functions, Linear Integer Arithmetic or
Bit-vectors

THIS TALK: Quantifier-free formulas over Extensional Arrays

Solving SMT with DPLL(T)

Methodology:

$$\underbrace{read(a,j) \neq read(b,i)}_{1} \wedge (\underbrace{a = b}_{2} \vee \underbrace{a = write(b,i,x)}_{3}) \wedge \underbrace{read(a,i) \neq x}_{4} \wedge \underbrace{j = i}_{5}$$

- SAT solver returns model $[1, 2, 4, 5]$

Solving SMT with DPLL(T)

Methodology:

$$\underbrace{read(a,j) \neq read(b,i)}_{1} \wedge (\underbrace{a = b}_{2} \vee \underbrace{a = write(b,i,x)}_{3}) \wedge \underbrace{read(a,i) \neq x}_{4} \wedge \underbrace{j = i}_{5}$$

- SAT solver returns model [1, 2, 4, 5]
- Theory solver says T-inconsistent

UPC
Departament de Llenguatges i Sistemes Informàtics
UNIVERSITAT POLITÈCNICA DE CATALUNYA

Solving SMT with DPLL(T)

Methodology:

$$\underbrace{read(a,j) \neq read(b,i)}_{1} \wedge \left(\underbrace{a=b}_{2} \vee \underbrace{a = write(b,i,x)}_{3}\right) \wedge \underbrace{read(a,i) \neq x}_{4} \wedge \underbrace{j=i}_{5}$$

- SAT solver returns model $[1, 2, 4, 5]$
- Theory solver says T-inconsistent
- Send $\{1,\ 2 \vee 3,\ 4,\ 5,\ \overline{1} \vee \overline{2} \vee \overline{4} \vee \overline{5}\}$ to SAT solver

Departament de Llenguatges i Sistemes Informatics
UNIVERSITAT POLITECNICA DE CATALUNYA

Solving SMT with DPLL(T)

Methodology:

$$\underbrace{read(a,j) \neq read(b,i)}_{1} \wedge (\underbrace{a = b}_{2} \vee \underbrace{a = write(b,i,x)}_{3}) \wedge \underbrace{read(a,i) \neq x}_{4} \wedge \underbrace{j = i}_{5}$$

- SAT solver returns model $[1, 2, 4, 5]$
- Theory solver says T-inconsistent
- Send $\{1,\ 2 \vee 3,\ 4,\ 5,\ \overline{1} \vee \overline{2} \vee \overline{4} \vee \overline{5}\}$ to SAT solver
- SAT solver returns model $[1, \overline{2}, 3, 4, 5]$

Solving SMT with DPLL(T)

Methodology:

$$\underbrace{read(a,j) \neq read(b,i)}_{1} \wedge \ (\underbrace{a = b}_{2} \vee \underbrace{a = write(b,i,x)}_{3}) \wedge \underbrace{read(a,i) \neq x}_{4} \wedge \underbrace{j = i}_{5}$$

- SAT solver returns model $[1, 2, 4, 5]$
- Theory solver says T-inconsistent
- Send $\{1,\ 2 \vee 3,\ 4,\ 5,\ \overline{1} \vee \overline{2} \vee \overline{4} \vee \overline{5}\}$ to SAT solver
- SAT solver returns model $[1, \overline{2}, 3, 4, 5]$
- Theory solver says T-inconsistent

Solving SMT with DPLL(T)

Methodology:

$$\underbrace{read(a,j) \neq read(b,i)}_{1} \wedge (\underbrace{a=b}_{2} \vee \underbrace{a = write(b,i,x)}_{3}) \wedge \underbrace{read(a,i) \neq x}_{4} \wedge \underbrace{j=i}_{5}$$

- SAT solver returns model $[1, 2, 4, 5]$
- Theory solver says T-inconsistent
- Send $\{1,\ 2 \vee 3,\ 4,\ 5,\ \overline{1} \vee \overline{2} \vee \overline{4} \vee \overline{5}\}$ to SAT solver
- SAT solver returns model $[1, \overline{2}, 3, 4, 5]$
- Theory solver says T-inconsistent
- SAT solver detects $\{1,\ 2 \vee 3,\ 4,\ 5,\ \overline{1} \vee \overline{2} \vee \overline{4} \vee \overline{5},\ \overline{1} \vee \overline{3} \vee \overline{4} \vee \overline{5}\}$
  UNSAT

UPC
Departament de Llenguatges i Sistemes Informatics
UNIVERSITAT POLITECNICA DE CATALUNYA

Solving SMT with DPLL(T)

Methodology:

$$\underbrace{read(a,j) \neq read(b,i)}_{1} \wedge (\underbrace{a=b}_{2} \vee \underbrace{a=write(b,i,x)}_{3}) \wedge \underbrace{read(a,i) \neq x}_{4} \wedge \underbrace{j=i}_{5}$$

- SAT solver returns model $[1, 2, 4, 5]$
- Theory solver says T-inconsistent
- Send $\{1,\ 2 \vee 3,\ 4,\ 5,\ \overline{1} \vee \overline{2} \vee \overline{4} \vee \overline{5}\}$ to SAT solver
- SAT solver returns model $[1, \overline{2}, 3, 4, 5]$
- Theory solver says T-inconsistent
- SAT solver detects $\{1,\ 2 \vee 3,\ 4,\ 5,\ \overline{1} \vee \overline{2} \vee \overline{4} \vee \overline{5},\ \overline{1} \vee \overline{3} \vee \overline{4} \vee \overline{5}\}$
  UNSAT

Two components: Boolean engine DPLL(X) + T-Solver

Solving SMT with DPLL(T) (2)

Several optimizations for enhancing efficiency:

- Check T-consistency only of full prop. models (at a leaf)

Solving SMT with DPLL(T) (2)

Several optimizations for enhancing efficiency:

- Ch~~eck T-consistency only of full prop. models (at a leaf)~~
- Check T-consistency of partial assignment while being built

Departament de Llenguatges i Sistemes Informàtics
UNIVERSITAT POLITÈCNICA DE CATALUNYA

Solving SMT with DPLL(T) (2)

Several optimizations for enhancing efficiency:

- C~~heck T-consistency only of full prop. models (at a leaf)~~
- Check T-consistency of partial assignment while being built
- Given a T-inconsistent assignment M, add $\neg M$ as a clause

Solving SMT with DPLL(T) (2)

Several optimizations for enhancing efficiency:

- ~~Check T-consistency only of full prop. models (at a leaf)~~
- Check T-consistency of partial assignment while being built

- ~~Given a T-inconsistent assignment M, add $\neg M$ as a clause~~
- Given a T-inconsistent assignment M, identify a T-inconsistent subset $M_0 \subseteq M$ and add $\neg M_0$ as a clause

Solving SMT with DPLL(T) (2)

Several optimizations for enhancing efficiency:

- ~~Check T-consistency only of full prop. models (at a leaf)~~
- Check T-consistency of partial assignment while being built
- ~~Given a T-inconsistent assignment M, add $\neg M$ as a clause~~
- Given a T-inconsistent assignment M, identify a T-inconsistent subset $M_0 \subseteq M$ and add $\neg M_0$ as a clause
- Upon a T-inconsistency, add clause and restart

Solving SMT with DPLL(T) (2)

Several optimizations for enhancing efficiency:

- ~~Check T-consistency only of full prop. models (at a leaf)~~
- Check T-consistency of partial assignment while being built

- ~~Given a T-inconsistent assignment M, add $\neg M$ as a clause~~
- Given a T-inconsistent assignment M, identify a T-inconsistent subset $M_0 \subseteq M$ and add $\neg M_0$ as a clause

- ~~Upon a T-inconsistency, add clause and restart~~
- Upon a T-inconsistency, bactrack to some point where the assignment was still T-consistent

Solving SMT with DPLL(T) (2)

Several optimizations for enhancing efficiency:

- ~~Check T-consistency only of full prop. models (at a leaf)~~
- Check T-consistency of partial assignment while being built

- ~~Given a T-inconsistent assignment M, add $\neg M$ as a clause~~
- Given a T-inconsistent assignment M, identify a T-inconsistent subset $M_0 \subseteq M$ and add $\neg M_0$ as a clause

- ~~Upon a T-inconsistency, add clause and restart~~
- Upon a T-inconsistency, bactrack to some point where the assignment was still T-consistent

> THIS TALK: obtain an *Arr*-solver that is incremental, backtrackable and produce inconsistency explanations

Solving SMT with DPLL(T) (3)

Need of case analysis inside the T-Solver:

$$\{\ \underbrace{write(a,i,x) = write(b,j,y)}_{1},\ \underbrace{write(c,i,x) \neq write(c,j,y)}_{2},\ \underbrace{read(a,j) \neq y}_{3}\ \}$$

It's inconsistent, but we need a case analysis on $i = j$

UPC
Departament de Llenguatges i Sistemes Informatics
UNIVERSITAT POLITECNICA DE CATALUNYA

Solving SMT with DPLL(T) (3)

Need of case analysis inside the T-Solver:

$$\{\ \underbrace{write(a,i,x) = write(b,j,y)}_{1},\ \underbrace{write(c,i,x) \neq write(c,j,y)}_{2},\ \underbrace{read(a,j) \neq y}_{3}\ \}$$

It's inconsistent, but we need a case analysis on $i = j$

- Assume $i = j$:
  From 1 we infer $x = y$
  From 2 we infer $x \neq y$ Inconsistency

UPC
Departament de Llenguatges i Sistemes Informatics
UNIVERSITAT POLITECNICA DE CATALUNYA

Solving SMT with DPLL(T) (3)

Need of case analysis inside the T-Solver:

$$\{ \underbrace{write(a,i,x) = write(b,j,y)}_{1},\ \underbrace{write(c,i,x) \neq write(c,j,y)}_{2},\ \underbrace{read(a,j) \neq y}_{3} \}$$

It's inconsistent, but we need a case analysis on $i = j$

- Assume $i = j$:
  From 1 we infer $x = y$
  From 2 we infer $x \neq y$ Inconsistency
- Assume $i \neq j$:
  From 1 we infer that a at position j has y
  which contradicts 3 Inconsistency

UPC
Departament de Llenguatges i Sistemes Informatics
UNIVERSITAT POLITECNICA DE CATALUNYA

Solving SMT with DPLL(T) (3)

Need of case analysis inside the T-Solver:

$$\{\ \underbrace{write(a,i,x) = write(b,j,y)}_{1},\ \underbrace{write(c,i,x) \neq write(c,j,y)}_{2},\ \underbrace{read(a,j) \neq y}_{3}\ \}$$

It's inconsistent, but we need a case analysis on $i = j$

- Assume $i = j$:
  From 1 we infer $x = y$
  From 2 we infer $x \neq y$ Inconsistency
- Assume $i \neq j$:
  From 1 we infer that a at position j has y
  which contradicts 3 Inconsistency

We use split-on-demand: case analysis done by the boolean engine

Overview of the talk

- SAT Modulo Theories (SMT)
  - The Theory of Extensional Arrays
  - Solving SMT with DPLL(T)
- **Handling Arrays in SMT**
  - Theory instantiation for Arrays
  - A new solver for the theory of Arrays
- Key points
- Experimental evaluation
- Conclusions

Handling Arrays in SMT

There are basically two possibilities:

- Using theory instantiation
- Having an *Arr*-solver for DPLL(*Arr*)

Departament de Llenguatges i Sistemes Informatics
UNIVERSITAT POLITECNICA DE CATALUNYA

Theory instantiation for Arrays

- There is no explicit T-Solver for Arrays
- Instead, have a Module that generate Lemmas
  Lemmas are instances of the axioms of the theory

Add the Lemmas to the set of clauses used by the SAT engine.

UPC
Departament de Llenguatges i Sistemes Informàtics
UNIVERSITAT POLITÈCNICA DE CATALUNYA

Theory instantiation for Arrays

- There is no explicit T-Solver for Arrays
- Instead, have a Module that generate Lemmas
  Lemmas are instances of the axioms of the theory

  Add the Lemmas to the set of clauses used by the SAT engine.

- Used in SMT solvers like Yices or Z3
- [Goel,Krstic&Fuch2008] studied completeness

Departament de Llenguatges i Sistemes Informàtics
UNIVERSITAT POLITÈCNICA DE CATALUNYA

Theory instantiation for Arrays

- There is no explicit T-Solver for Arrays
- Instead, have a Module that generate Lemmas
  Lemmas are instances of the axioms of the theory

  Add the Lemmas to the set of clauses used by the SAT engine.

- Used in SMT solvers like Yices or Z3
- [Goel,Krstic&Fuch2008] studied completeness

- Positive: simple and easier to implement
- Negative: cannot use dedicated algorithms for the Theory

Theory instantiation for Arrays(2)

To see pros and cons

Consider a simpler theory: uninterpreted funtions

Departament de Llenguatges i Sistemes Informatics
UNIVERSITAT POLITEČNICA DE CATALUNYA

Theory instantiation for Arrays(2)

To see pros and cons

Consider a simpler theory: uninterpreted funtions

- Using Theory Instantiation:
  Generate Lemmas like

$$a = b \Rightarrow fa = fb$$

if f is a function symbol and a and b are constants.

UPC
Departament de Llenguatges i Sistemes Informatics
UNIVERSITAT POLITECNICA DE CATALUNYA

Theory instantiation for Arrays(2)

To see pros and cons

Consider a simpler theory: uninterpreted funtions

- Using Theory Instantiation:
  Generate Lemmas like

$$a = b \Rightarrow fa = fb$$

  if f is a function symbol and a and b are constants.

- Having a T-Solver:
  Apply congruence closure on the set of equality literals.

Theory instantiation for Arrays(2)

To see pros and cons

Consider a simpler theory: uninterpreted funtions

- Using Theory Instantiation:
  Generate Lemmas like

$$a = b \Rightarrow fa = fb$$

  if f is a function symbol and a and b are constants.

- Having a T-Solver:
  Apply congruence closure on the set of equality literals.

It's not obvious what's the best

Departament de Llenguatges i Sistemes Informatics
UNIVERSITAT POLITECNICA DE CATALUNYA

Theory instantiation for Arrays(2)

To see pros and cons

Consider a simpler theory: uninterpreted funtions

- Using Theory Instantiation:
  Generate Lemmas like

$$a = b \Rightarrow fa = fb$$

  if f is a function symbol and a and b are constants.
- Having a T-Solver:
  Apply congruence closure on the set of equality literals.

It's not obvious what's the best

We believe that the same happens with the Theory of Arrays

A new solver for the Theory of Arrays

A new solver for the Theory of Arrays

- Existing Solver [Stump,Barrett,Dill&Levitt2001]:

  Based on the “read” operator

  We call it Read-based:

  write operators are translated into read operators.

A new solver for the Theory of Arrays

- Existing Solver [Stump,Barrett,Dill&Levitt2001]:

  Based on the "read" operator

  We call it Read-based:

  write operators are translated into read operators.

- New approach:

  We call it Write-based:

  read operators are translated into write operators.

A new solver for the Theory of Arrays(2)

- Read-based:

$$a = write(b, i, x)$$

Departament de Llenguatges i Sistemes Informatics
UNIVERSITAT POLITECNICA DE CATALUNYA

A new solver for the Theory of Arrays(2)

- Read-based:

$$a = write(b, i, x)$$

$$\Downarrow$$

is translated into

UPC
Departament de Llenguatges i Sistemes Informatics
UNIVERSITAT POLITECNICA DE CATALUNYA

A new solver for the Theory of Arrays(2)

- Read-based:

$$a = \textit{write}(b, i, x)$$

$$\Downarrow$$

$$\textit{read}(a, i) = x$$

$$+$$

is translated into

A new solver for the Theory of Arrays(2)

- Read-based:

$$a = write(b, i, x)$$

$$\Downarrow \qquad\qquad \text{is translated into}$$

$$read(a, i) = x$$

$$+$$

$$a \simeq b \qquad\qquad ???$$

UPC
Departament de Llenguatges i Sistemes Informàtics
UNIVERSITAT POLITÈCNICA DE CATALUNYA

A new solver for the Theory of Arrays(2)

- Read-based:

$$a = write(b, i, x)$$

$$\Downarrow \qquad \text{is translated into}$$

$$read(a, i) = x$$

$$+$$

$$a =_i b \qquad \text{equal except in } i$$

Departament de Llenguatges i Sistemes Informatics
UNIVERSITAT POLITECNICA DE CATALUNYA

A new solver for the Theory of Arrays(2)

- Read-based:

$$a = write(b, i, x)$$

$$\Downarrow \qquad \text{is translated into}$$

$$read(a, i) = x$$

$$+$$

$$a =_i b \qquad \text{equal except in } i$$

Basically, ends up with uniterpreted funtions
plus this new theory of I-equality of arrays
(which can be handled using theory instantiation)

UPC
Departament de Llenguatges i Sistemes Informatics
UNIVERSITAT POLITECNICA DE CATALUNYA

A new solver for the Theory of Arrays(2)

- Read-based:

$$a = write(b, i, x)$$
$$\Downarrow \qquad \text{is translated into}$$
$$read(a, i) = x$$
$$+$$
$$a =_i b \qquad \text{equal except in } i$$

- Write-based:

$$read(a, i) = x$$

Departament de Llenguatges i Sistemes Informàtics
UNIVERSITAT POLITÈCNICA DE CATALUNYA

A new solver for the Theory of Arrays(2)

- Read-based:

$$a = write(b, i, x)$$

$$\Downarrow$$

$$read(a, i) = x$$

$$+$$

$$a =_i b$$

is translated into

equal except in i

- Write-based:

$$read(a, i) = x$$

$$\Downarrow$$

Departament de Llenguatges i Sistemes Informatics
UNIVERSITAT POLITÈCNICA DE CATALUNYA

A new solver for the Theory of Arrays(2)

- Read-based:

$$a = write(b, i, x)$$
$$\Downarrow \qquad \text{is translated into}$$
$$read(a, i) = x$$
$$+$$
$$a =_i b \qquad \text{equal except in } i$$

- Write-based:

$$read(a, i) = x$$
$$\Downarrow$$
$$a = write(b, i, x) \qquad \text{for some fresh } b$$

Departament de Llenguatges i Sistemes Informatics
UNIVERSITAT POLITECNICA DE CATALUNYA

A new solver for the Theory of Arrays(2)

- Read-based:

$$a = write(b, i, x)$$
$$\Downarrow \qquad \text{is translated into}$$
$$read(a, i) = x$$
$$+$$
$$a =_i b \qquad \text{equal except in } i$$

- Write-based:

$$read(a, i) = x$$
$$\Downarrow$$
$$a = write(b, i, x) \qquad \text{for some fresh } b$$

We follow the Write-based approach

UPC
Departament de Llenguatges i Sistemes Informatics
UNIVERSITAT POLITECNICA DE CATALUNYA

A new solver for the Theory of Arrays(3)

Set of literals:

$$a = write(b, j, x)$$
$$b = write(c, i, y)$$
$$d = write(e, i, y)$$
$$a = d$$

UPC
Departament de Llenguatges i Sistemes Informatics
UNIVERSITAT POLITECNICA DE CATALUNYA

A new solver for the Theory of Arrays(3)

Set of literals:

$$a = write(b, j, x)$$
$$b = write(c, i, y)$$
$$d = write(e, i, y)$$
$$a = d$$

Representation:

a	j	x
b	i	y
c		

=

d	i	y
e		

A new solver for the Theory of Arrays(3)

Set of literals:

$$a = write(b, j, x)$$
$$b = write(c, i, y)$$
$$d = write(e, i, y)$$
$$a = d$$

Representation:

a	j	x
b	i	y
c		

=

d	i	y
e		

Which "writes" are relevant?

Departament de Llenguatges i Sistemes Informatics
UNIVERSITAT POLITECNICA DE CATALUNYA

A new solver for the Theory of Arrays(3)

Set of literals:

$$a = write(b, j, x)$$
$$b = write(c, i, y)$$
$$d = write(e, i, y)$$
$$a = d$$

Representation:

a	j	x
b	i	y
c		

=

d	i	y
e		

Which “writes” are relevant?

- if $i = j$ then we need $x = y$

A new solver for the Theory of Arrays(3)

Set of literals:

$a = write(b, j, x)$
$b = write(c, i, y)$
$d = write(e, i, y)$
$a = d$

Representation:

a	j	x
b	i	y
c		

=

d	i	y
e		

Which "writes" are relevant?

- if $i = j$ then we need $x = y$
- if $i \neq j$ we need $e = write(e_1, j, x)$

A new solver for the Theory of Arrays(3)

Set of literals:

$$a = write(b, j, x)$$
$$b = write(c, i, y)$$
$$d = write(e, i, y)$$
$$a = d$$

Representation:

a	j	x
b	i	y
c		

=

d	i	y
e		

Which “writes” are relevant?

- if $i = j$ then we need $x = y$
- if $i \neq j$ we need $e = write(e_1, j, x)$

A new solver for the Theory of Arrays(3)

Set of literals:

$$a = write(b, j, x)$$
$$b = write(c, i, y)$$
$$d = write(e, i, y)$$
$$a = d$$

Representation:

a	j	x
b	i	y
c		

=

d	i	y
e	j	x
e1		

Which “writes” are relevant?

- if $i = j$ then we need $x = y$
- if $i \neq j$ we need $e = write(e_1, j, x)$

A new solver for the Theory of Arrays(3)

Set of literals:

$a = write(b, j, x)$
$b = write(c, i, y)$
$d = write(e, i, y)$
$a = d$

Representation:

a	j	x
b	i	y
c		

=

d	i	y
e	j	x
e1		

Which "writes" are relevant?

- if $i = j$ then we need $x = y$
- if $i \neq j$ we need $e = write(e_1, j, x)$

Recall: we may need splitting on $i = j$

Overview of the talk

- SAT Modulo Theories (SMT)
  - The Theory of Extensional Arrays
  - Solving SMT with DPLL(T)
- Handling Arrays in SMT
  - Theory instantiation for Arrays
  - A new solver for the theory of Arrays
- Key points
- Experimental evaluation
- Conclusions

Key points

There are three key points in our approach:

Key points

There are three key points in our approach:

- Notion of solved form:
  Early detection of satisfiable sets of literals

Departament de Llenguatges i Sistemes Informatics
UNIVERSITAT POLITÈCNICA DE CATALUNYA

Key points

There are three key points in our approach:

- Notion of solved form:
  Early detection of satisfiable sets of literals
- Delay negative witnesses introduction:

Key points

There are three key points in our approach:

- Notion of solved form:
  Early detection of satisfiable sets of literals
- Delay negative witnesses introduction:
  Recall the extensionality axiom:

$$a \neq b \Rightarrow \exists i. read(a,i) \neq read(b,i)$$

Key points

There are three key points in our approach:

- Notion of solved form:
  Early detection of satisfiable sets of literals
- Delay negative witnesses introduction:

$$a \neq b$$
$$\Downarrow$$
$$a = write(a_1, ni, ne_1) \text{ and } b = write(b_2, ni, ne_2)$$

  where ni is a new index and ne_1 and ne_2 are fresh constants with $ne_1 \neq ne_2$

Key points

There are three key points in our approach:

- Notion of solved form:
  Early detection of satisfiable sets of literals
- Delay negative witnesses introduction:

$$a \neq b$$
$$\Downarrow$$
$$a = write(a_1, ni, ne_1) \text{ and } b = write(b_2, ni, ne_2)$$

where ni is a new index and ne_1 and ne_2 are fresh constants with $ne_1 \neq ne_2$

This name is a tribute to Monty Python's "Ni knights" (check Google with "Knights who say Ni" for further details)

The relationship between them is that
both Ni's (the indexes and the Knights) introduce a lot of noise

Departament de Llenguatges i Sistemes Informàtics
UNIVERSITAT POLITÈCNICA DE CATALUNYA

Key points

There are three key points in our approach:

- Notion of solved form:
  Early detection of satisfiable sets of literals
- Delay negative witnesses introduction:
  Delay the introduction of “Ni’s” avoiding unnecessary case analisys

Key points

There are three key points in our approach:

- Notion of solved form:
  Early detection of satisfiable sets of literals
- Delay negative witnesses introduction:
  Delay the introduction of "Ni's" avoiding unnecessary case analisys
- Produce better(shorter) explanations:
  Using specialized mechanisms that take into account the knowledge about the theory of Arrays

Key points: Solved forms

There are several solved situations

Three particular examples (see paper for general definition):

Key points: Solved forms

There are several solved situations

Three particular examples (see paper for general definition):

- $write(a, i, x) = write(b, j, y)$
  if $i = j$, $x = y$ and a and b are different free constants.

Departament de Llenguatges i Sistemes Informàtics
UNIVERSITAT POLITÈCNICA DE CATALUNYA

Key points: Solved forms

There are several solved situations

Three particular examples (see paper for general definition):

- $write(a,i,x) = write(b,j,y)$
  if $i = j$, $x = y$ and a and b are different free constants.

- $write(a,i,x) \neq write(b,j,y)$
  if we don't have $i = j$ and b is a free constant.

Key points: Solved forms

There are several solved situations

Three particular examples (see paper for general definition):

- $write(a,i,x) = write(b,j,y)$
  if $i = j, x = y$ and a and b are different free constants.
- $write(a,i,x) \neq write(b,j,y)$
  if we don't have $i = j$ and b is a free constant.
- $write(a,i,x) \neq write(b,i,y)$
  if we have neither $x = y$ nor $x \neq y$.

UPC
Departament de Llenguatges i Sistemes Informatics
UNIVERSITAT POLITECNICA DE CATALUNYA

Key points: Solved forms(2)

We can complete our partial model as follows:

Departament de Llenguatges i Sistemes Informatics
UNIVERSITAT POLITECNICA DE CATALUNYA

Key points: Solved forms(2)

We can complete our partial model as follows:

- Indexes and values:
  $\forall v_1$ and v_2, if neither $v_1 = v_2$ nor $v_2 \neq v_1$ in the partial model we take $v_2 \neq v_1$.

Key points: Solved forms(2)

We can complete our partial model as follows:

- Indexes and values:
  $\forall v_1$ and v_2, if neither $v_1 = v_2$ nor $v_2 \neq v_1$ in the partial model we take $v_2 \neq v_1$.
- Arrays: assume there is a value d different from all others.
  $\forall$ array A, if $A[i]$ is not defined for some i in the partial model we take $A[i] = d$

UPC
Departament de Llenguatges i Sistemes Informatics
UNIVERSITAT POLITECNICA DE CATALUNYA

Key points: Solved forms(2)

We can complete our partial model as follows:

- Indexes and values:
  $\forall v_1$ and v_2, if neither $v_1 = v_2$ nor $v_2 \neq v_1$ in the partial model we take $v_2 \neq v_1$.
- Arrays: assume there is a value d different from all others.
  $\forall$ array A, if $A[i]$ is not defined for some i in the partial model we take $A[i] = d$
  - $write(a,i,x) = write(b,j,y)$
    if $i = j$, $x = y$ and a and b are different free constants.

Departament de Llenguatges i Sistemes Informàtics
UNIVERSITAT POLITÈCNICA DE CATALUNYA

Key points: Solved forms(2)

We can complete our partial model as follows:

- Indexes and values:
  $\forall v_1$ and v_2, if neither $v_1 = v_2$ nor $v_2 \neq v_1$ in the partial model we take $v_2 \neq v_1$.
- Arrays: assume there is a value d different from all others.
  $\forall$ array A, if $A[i]$ is not defined for some i in the partial model we take $A[i] = d$
  - $write(a,i,x) = write(b,j,y)$
    if $i = j$, $x = y$ and a and b are different free constants.
    Since a and b are free constants
    they have the same interpretation in the model.

UPC
Departament de Llenguatges i Sistemes Informatics
UNIVERSITAT POLITECNICA DE CATALUNYA

Key points: Solved forms(2)

We can complete our partial model as follows:

- Indexes and values:
  $\forall v_1$ and v_2, if neither $v_1 = v_2$ nor $v_2 \neq v_1$ in the partial model
  we take $v_2 \neq v_1$.
- Arrays: assume there is a value d different from all others.
  $\forall$ array A, if $A[i]$ is not defined for some i in the partial model
  we take $A[i] = d$
  - $write(a,i,x) = write(b,j,y)$
    if $i = j$, $x = y$ and a and b are different free constants.
    Since a and b are free constants
    they have the same interpretation in the model.
    which satisfies the literal

UPC
Departament de Llenguatges i Sistemes Informàtics
UNIVERSITAT POLITÈCNICA DE CATALUNYA

Key points: Solved forms(2)

We can complete our partial model as follows:

- Indexes and values:
  $\forall v_1$ and v_2, if neither $v_1 = v_2$ nor $v_2 \neq v_1$ in the partial model we take $v_2 \neq v_1$.
- Arrays: assume there is a value d different from all others.
  $\forall$ array A, if $A[i]$ is not defined for some i in the partial model we take $A[i] = d$
  - $write(a,i,x) \neq write(b,j,y)$
    if we don't have $i = j$ and b is a free constant.

UPC
Departament de Llenguatges i Sistemes Informàtics
UNIVERSITAT POLITÈCNICA DE CATALUNYA

Key points: Solved forms(2)

We can complete our partial model as follows:

- Indexes and values:
  $\forall v_1$ and v_2, if neither $v_1 = v_2$ nor $v_2 \neq v_1$ in the partial model we take $v_2 \neq v_1$.
- Arrays: assume there is a value d different from all others.
  $\forall$ array A, if $A[i]$ is not defined for some i in the partial model we take $A[i] = d$
  - $write(a,i,x) \neq write(b,j,y)$
    if we don't have $i = j$ and b is a free constant.
    Since we don't have $i = j$
    we take $i \neq j$ in the model and

UPC
Departament de Llenguatges i Sistemes Informatics
UNIVERSITAT POLITEČNICA DE CATALUNYA

Key points: Solved forms(2)

We can complete our partial model as follows:

- Indexes and values:
  $\forall v_1$ and v_2, if neither $v_1 = v_2$ nor $v_2 \neq v_1$ in the partial model
  we take $v_2 \neq v_1$.
- Arrays: assume there is a value d different from all others.
  $\forall$ array A, if $A[i]$ is not defined for some i in the partial model
  we take $A[i] = d$
  - $write(a, i, x) \neq write(b, j, y)$
    if we don't have $i = j$ and b is a free constant.
    Since we don't have $i = j$
    we take $i \neq j$ in the model and
    since b is free constant
    we take $b[i] = d \neq x$ in the model

Key points: Solved forms(2)

We can complete our partial model as follows:

- Indexes and values:
  $\forall v_1$ and v_2, if neither $v_1 = v_2$ nor $v_2 \neq v_1$ in the partial model
  we take $v_2 \neq v_1$.
- Arrays: assume there is a value d different from all others.
  $\forall$ array A, if $A[i]$ is not defined for some i in the partial model
  we take $A[i] = d$
  - $write(a,i,x) \neq write(b,j,y)$
    if we don't have $i = j$ and b is a free constant.
    Since we don't have $i = j$
    we take $i \neq j$ in the model and
    since b is free constant
    we take $b[i] = d \neq x$ in the model
    which satisfies the literal

Key points: Solved forms(2)

We can complete our partial model as follows:

- Indexes and values:
  $\forall v_1$ and v_2, if neither $v_1 = v_2$ nor $v_2 \neq v_1$ in the partial model we take $v_2 \neq v_1$.
- Arrays: assume there is a value d different from all others.
  $\forall$ array A, if $A[i]$ is not defined for some i in the partial model we take $A[i] = d$
  - $write(a,i,x) \neq write(b,i,y)$
    if we have neither $x = y$ nor $x \neq y$.

UPC
Departament de Llenguatges i Sistemes Informàtics
UNIVERSITAT POLITÈCNICA DE CATALUNYA

Key points: Solved forms(2)

We can complete our partial model as follows:

- Indexes and values:
  $\forall v_1$ and v_2, if neither $v_1 = v_2$ nor $v_2 \neq v_1$ in the partial model
  we take $v_2 \neq v_1$.
- Arrays: assume there is a value d different from all others.
  $\forall$ array A, if $A[i]$ is not defined for some i in the partial model
  we take $A[i] = d$
  - $write(a,i,x) \neq write(b,i,y)$
    if we have neither $x = y$ nor $x \neq y$.
    since we have neither $x = y$ nor $x \neq y$
    we take $x \neq y$ in the model

Key points: Solved forms(2)

We can complete our partial model as follows:

- Indexes and values:
  $\forall v_1$ and v_2, if neither $v_1 = v_2$ nor $v_2 \neq v_1$ in the partial model
  we take $v_2 \neq v_1$.
- Arrays: assume there is a value d different from all others.
  $\forall$ array A, if $A[i]$ is not defined for some i in the partial model
  we take $A[i] = d$
  - $write(a,i,x) \neq write(b,i,y)$
    if we have neither $x = y$ nor $x \neq y$.
    since we have neither $x = y$ nor $x \neq y$
    we take $x \neq y$ in the model
    which satisfies the literal

UPC
Departament de Llenguatges i Sistemes Informàtics
UNIVERSITAT POLITÈCNICA DE CATALUNYA

Key points: Solved forms(2)

We can complete our partial model as follows:

- Indexes and values:
  $\forall v_1$ and v_2, if neither $v_1 = v_2$ nor $v_2 \neq v_1$ in the partial model we take $v_2 \neq v_1$.
- Arrays: assume there is a value d different from all others.
  $\forall$ array A, if $A[i]$ is not defined for some i in the partial model we take $A[i] = d$

We have several inference rules that transform literals NOT in solved form until they are (see paper for details).

Key points: Delay Ni's introduction

Consider the following negative literal:

a1	i1	x1
a2	i2	x2
a3		

$\neq$

b1	i2	y2
b2	i1	x1
b3		

With: $i_1 \neq i_2 \;\; \wedge \;\; x_2 \neq y_2$

Key points: Delay Ni's introduction

Consider the following negative literal:

a1	i1	x1
a2	i2	x2
a3		

$\neq$

b1	i2	y2
b2	i1	x1
b3		

With: $i_1 \neq i_2 \;\; \wedge \;\; x_2 \neq y_2$

There is no need to add any new index *ni*
Avoiding case analysis between *ni* and the other indexes.

Departament de Llenguatges i Sistemes Informàtics
UNIVERSITAT POLITÈCNICA DE CATALUNYA

Key points: Delay Ni's introduction(2)

Consider the following negative literal:

a1	i1	x1
a2	i2	x2
a3		

$\neq$

b1	i2	x2
b2	i1	x1
b3		

With: $i_1 \neq i_2$

Key points: Delay Ni's introduction(2)

Consider the following negative literal:

a1	i1	x1
a2	i2	x2
a3		

$\neq$

b1	i2	x2
b2	i1	x1
b3		

With: $i_1 \neq i_2$

We have to add a new index *ni*, but we add it at the end.

$a_3 = write(a_4, ni, ed_1) \ \wedge \ b_3 = write(b_4, ni, ed_2)$
with $ed_1 \neq ed_2$, $ni \neq i_1$ and $ni \neq i_2$

Key points: Delay Ni's introduction(2)

Consider the following negative literal:

a1	i1	x1
a2	i2	x2
a3		

$\neq$

b1	i2	x2
b2	i1	x1
b3		

With: $i_1 \neq i_2$

We have to add a new index *ni*, but we add it at the end.

$a_3 = write(a_4, ni, ed_1) \ \wedge \ b_3 = write(b_4, ni, ed_2)$
with $ed_1 \neq ed_2$, $ni \neq i_1$ and $ni \neq i_2$

a1	i1	x1
a2	i2	x2
a3	ni	ed1
a4		

$\neq$

b1	i2	x2
b2	i1	x1
b3	ni	ed2
b4		

Key points: Shorter explanations

Consider the following incosistent literal
with $i_1 \neq i_3 \;\wedge\; i_2 \neq i_3 \;\wedge\; i_1 \neq i_2$:

a1	i1	x1
a2	i2	x2
a3	i3	x3
c		

$\neq$

b1	i3	x3
b2	i1	x1
b3	i2	x2
c		

Inconsistency explanation: $a_1 \neq b_1 \;\wedge\; i_1 \neq i_3 \;\wedge\; i_2 \neq i_3 \;\wedge\; i_1 \neq i_2$

Key points: Shorter explanations

Consider the following incosistent literal
with $i_1 \neq i_3 \ \wedge \ i_2 \neq i_3 \ \wedge \ i_1 \neq i_2$:

a1	i1	x1
a2	i2	x2
a3	i3	x3
c		

$\neq$

b1	i3	x3
b2	i1	x1
b3	i2	x2
c		

Inconsistency explanation: $a_1 \neq b_1 \ \wedge \ i_1 \neq i_3 \ \wedge \ i_2 \neq i_3 \ \wedge \ i_1 \neq i_2$

Key points: Shorter explanations

Consider the following incosistent literal
with $i_1 \neq i_3 \;\wedge\; i_2 \neq i_3 \;\wedge\; i_1 \neq i_2$:

a1	i1	x1
a2	i2	x2
a3	i3	x3
c		

$\neq$

b1	i3	x3
b2	i1	x1
b3	i2	x2
c		

Inconsistency explanation: $a_1 \neq b_1 \;\wedge\; i_1 \neq i_3 \;\wedge\; i_2 \neq i_3 \;\wedge\; \cancel{i_1 \neq i_2}$

UPC
Departament de Llenguatges i Sistemes Informatics
UNIVERSITAT POLITECNICA DE CATALUNYA

Overview of the talk

- SAT Modulo Theories (SMT)
  - The Theory of Extensional Arrays
  - Solving SMT with DPLL(T)
- Handling Arrays in SMT
  - Theory instantiation for Arrays
  - A new solver for the theory of Arrays
- Key points
- **Experimental evaluation**
- Conclusions

Experimental evaluation

Setting used: SMT-LIB benchmarks 2007, 300 sec.

	YICES 1.0.10		YICES 1.0		Z3 0.1		CVC3 1.2		BARCELOGIC	
	Tot	Max	Tot	Max	Tot	Max	Total	Max	Tot	Max
array_ben	52	42	69	52	21	8	496 (16)	294	282	162
cvc	5	4	4	3	1	1	114	57	59	38
qlock2	49	5	50	6	114	37	199 (30)	117	652	55
storecomm	35	0.1	41	0.1	37	0.1	993	20	48	0.1
storeinv	1	0.1	1	0.1	8	0.3	691 (162)	76	22	2
swap	970	130	581	60	1431	128	13726 (1263)	275	275	9

SMT competition 2008 results.

QF_AX:	Barcelogic winner.	Z3.2 second.	NO Timeouts.
QF_AUFLIA:	Z3.2 winner.	Barcelogic second.	NO Timeouts.

Departament de Llenguatges i Sistemes Informatics
UNIVERSITAT POLITECNICA DE CATALUNYA

Conclusions

- Our solver is intuitive and still competitive.
- Completely different from previous approaches.
- Observation: there is no unique best approach.

The more approaches we have the better

- Need of new hard benchmarks to compare and improve.

Thank you!

Departament de Llenguatges i Sistemes Informatics
UNIVERSITAT POLITECNICA DE CATALUNYA