Schemata of formulæ in the theory of arrays

Nicolas Peltier

Laboratoire d'Informatique de Grenoble/CNRS
CAPP team - ASAP project (ANR-09-BLAN-0407-01)

TABLEAUX 2013 - September 2013 - Nancy

The theory of arrays

Model array data-structures with update and retrieval operations:

- Two function symbols: *select* and *store*
- Two axioms:

$$select(store(t, i, v), i) = v$$

$$i \neq j \Rightarrow select(store(t, i, v), j) = select(t, j)$$

- Decidable (for quantified-free formulæ)

Very well-studied, numerous extensions (arithmetic...)

Schemata of formulæ

Logical formulæ parameterized by natural numbers:

- Arithmetic variables and constants
- Indexed signature: $p_0, q_{n+1}, \ldots$
- Generalized connectives: $\bigvee_{i=a}^{b} \phi$ and $\bigwedge_{i=a}^{b} \phi$, where a, b are arithmetic expressions

Example:

$$p_0 \wedge \bigwedge_{i=0}^{n} p_i \Rightarrow p_{i+1} \wedge \neg p_{n+1}$$

Known results in propositional logic

- Satisfiability is not decidable in general
- Decidable if the arithmetic expressions are "sufficiently simple" k or $n + k$, where n is variable, k a natural number

These results extend to other theories (TABLEAUX'11)

Schemata modulo the theory of arrays

Motivations

- Interesting from a theoretical point of view
- Potential applications in verification

  Schemata can be used to model programs with loops

  t_i = value of t at iteration rank i

Plan of the talk

1. The logic: syntax and semantics
2. Undecidability results
3. A decidable class
4. A proof procedure
5. Example of applications
6. Conclusion

Syntax

- A set of indexed or non-indexed constant symbols (with types $\mathtt{s}$ or $\mathtt{s} \to \mathtt{s}'$)

  $a_0,\ t_n,\ f_{n+1},\ \ldots$
- Arithmetic (non-indexed) symbols: $n, i : \mathbb{N}$
- Functions *select* and *store*, of profiles:

  $\mathit{select} : \mathtt{s} \to \mathtt{s}', \mathtt{s} \to \mathtt{s}'$

  $\mathit{store} : \mathtt{s} \to \mathtt{s}', \mathtt{s}, \mathtt{s}' \to \mathtt{s} \to \mathtt{s}'$
- Usual connectives $\vee$, $\wedge$, $\neg$
- Iterated connectives $\bigvee_{i=a}^{b} \psi$ and $\bigwedge_{i=a}^{b} \psi$, where ψ is a formula parameterized by i
- Arithmetic expressions of the form k or $n + k$ with $k \in \mathbb{N}$

Semantics

- Arithmetic constants are interpreted as natural numbers
- Other base objects are interpreted arbitrarily
- Objects of type $\mathtt{s} \rightarrow \mathtt{s}'$ are interpreted as arrays (or functions) from $\mathtt{s}$ to $\mathtt{s}'$
- *select* and *store* are interpreted as the usual retrieval and storage operations
  - $\mathit{select}(f, a) \rightarrow f(a)$
  - $\mathit{store}(f, i, a)$: function g such that $g(x) = a$ if $x = i$ and $g(x) = f(x)$ if $x \neq i$
- $\bigwedge_{i=a}^{b} \phi$ holds in I iff $\phi\{i \leftarrow k\}$ holds for every $k \in [I(a), I(b)]$

Syntax (alternative)

- Do not use the function *store*
- Use instead expressions of the form $t =_E s$, where E is a finite set of terms
- $t =_E s$ holds iff t and s agree on every element distinct from those in E

$$t = store(s, i, v) \Leftrightarrow select(t, i) = v \wedge t =_{\{i\}} s$$

Advantage: $=_E$-expressions can easily be combined:

$$x =_E y \wedge y =_{E'} z \Rightarrow x =_{E \cup E'} z$$

Syntax (restriction)

Additional restrictions

A unique free arithmetic variable (the parameter)
No nested iteration: $\bigvee_{i=0}^{n} \bigwedge_{j=0}^{n} (p_j \vee q_i)$ forbidden
All arithmetic expressions are of the form 0, i, $i+1$

Expressive power is not reduced (FI'13, JAIR'11)

e.g. $a_{i+2} = b_i$ is written $a'_{i+1} = b_i$ with the axiom:
$\bigwedge_{i=0}^{n} a'_{i+1} = a_i$
$\bigwedge_{i=1}^{n} \phi$ is written $\bigwedge_{i=0}^{n} (p_i \vee \phi) \wedge p_0 \wedge \bigvee_{i=0}^{n} \neg p_{i+1}$

Limitations

Satisfiability is *undecidable*

- Follows immediately from previous results (TABLEAUX'11)
- Known undecidability result: schemata of quantifier-free equational formulæ
- Can be encoded in the theory of arrays in a straightforward manner (arrays $\leftrightarrow$ functions)
- The encoding does not use the "store" operation

A new hope ?

Impose additional conditions on the arrays ?

- Forbid arrays of sort $\mathtt{s} \to \mathtt{s}$
  i.e., indices cannot be elements of the array
- The condition must hold also for derived arrays:
  $t_1 : \mathtt{s} \to \mathtt{s}'$ $\quad$ $t_2 : \mathtt{s}' \to \mathtt{s}$
  $t_2 \circ t_1 : \mathtt{s} \to \mathtt{s}$: forbidden
- More generally: there must exist an ordering $\prec$ s.t.:
  $t : \mathtt{s} \to \mathtt{s}' \Rightarrow \mathtt{s} \prec \mathtt{s}'$

A new hope ? (2)

Still undecidable

- Proof: encode the Post correspondence problem
- Let (w_j^i) with $j \in [1, n]$ and $i = 1, 2$ be two sequences of words
- Let $j_1, \ldots, j_k$ be a potential solution
- Store the words $w_{j_1}^i \ldots . w_{j_k}^i$ in an array W_i
- $\mathit{select}(W_i, a_j) = (m, l)$ if the j-th character of the word $w_{j_1}^i \ldots . w_{j_k}^i$ is the l-th character of the word w_m^i

Encoding of the Post correspondence problem

Many properties can be encoded straightforwardly:

- W_1 and W_2 really encode sequences of words:
  - $\bigwedge_{j=0}^{n} select(W_i, a_j) = (m, l) \Rightarrow select(W_i, a_{j+1}) = (m, l+1)$ if l is not the last character in w_m^i
  - $\bigwedge_{j=0}^{n} select(W_i, a_j) = (m, l) \Rightarrow \exists m'\, select(W_i, a_{j+1}) = (m', 1)$ if l is the last character in w_m^i
- $\exists m\, select(W_0^i, a_0) = (m, 1)$: the sequence starts at the beginning of a word
- W_1 and W_2 are identical:
  $\bigwedge_{j=0}^{n} select(W_1, a_j) \neq (m, l) \wedge select(W_2, a_j) \neq (m', l')$
  if character l of the word w_m^1 is distinct from character l' of the word $w_{m'}^2$

Note: theory of arrays useless

Encoding of the Post correspondence problem

We have encoded the facts that:

$$W^1 = w^1_{j_1} \ldots . w^1_{j_p}$$

$$W^2 = w^2_{j'_1} \ldots . w^2_{j'_q}$$

$$W^1 = W^2$$

More difficult: ensure that $(j_1, \ldots, j_p) = (j'_1, \ldots, j'_q)$

Encoding of the Post correspondence problem

- Use the axioms of the theory of arrays
- Use an indexed constant b_j^i to mark the cells corresponding to the beginning of a word in W_i
- State that the two sequences of words are identical:

$$\bigwedge_{i=0}^{n} select(W_1, b_i^1) = select(W_2, b_i^2)$$

Encoding of the Post correspondence problem

How can we ensure that $b_1^i, \ldots, b_n^i$ really correspond to the beginning of a word ?

- Define an array A^i that is true exactly at the cells corresponding to the start of a new word
  $\bigwedge_{j=1}^{n} \mathit{select}(A^i, a_i) = \mathtt{true} \Leftrightarrow (\exists m\, \mathit{select}(W^i, a_i) = (m, 1))$.
- Use the theory of arrays to ensure that A^i is true *exactly* at $b_1^i, \ldots, b_n^i$:
  - $\bigwedge_{i=0}^{n} \mathit{select}(B_0, a_j) = \mathtt{false}$
  - $\bigwedge_{j=1}^{n} B_{j+1} = \mathit{store}(B_j, b_j^i, \mathtt{true})$
  - $B_{n+1} = A^i$, for $i = 1, 2$

Decidability results

Further restrictions are needed

- Impose additional conditions on the *interpretations*
- If $a_i = b_j$ with $i < j$ then:
  - Either $a_i = c$, for some non-indexed constant c
  - Or $a_i = d_{i+1}$, for some indexed constant d

    i.e. $a_i = d_{i+1} = e_{i+2} = \ldots = b_j$
- This condition must hold for every sort $\mathtt{s}$ such that one of the following conditions hold:
  - The signature contains an array of sort $\mathtt{s} \to \mathtt{s}$ (possibly modulo array composition)
  - A store operation is performed on an array of sort $\mathtt{s} \to \mathtt{s}'$

An interpretation satisfying the restriction above is called *contiguous*.

Proof procedure

Basic ideas

1. Given a formula ϕ of parameter n, consider the two branches:

$$\phi \wedge n = 0 \qquad \phi \wedge n > 0$$

2. The first branch can be handled by usual decision procedures
3. For the second branch:
   - Replace n by $n+1$
   - Simplify the formula:
     - Propositional decomposition rules
     - Equality reasoning rules
     - Expand iterated connectives

$$\bigwedge_{i=0}^{n+1} \phi \rightarrow \bigwedge_{i=0}^{n} \phi \wedge \phi[n+1/i] \qquad \bigvee_{i=0}^{n+1} \phi \rightarrow \bigvee_{i=0}^{n} \phi \vee \phi[n+1/i]$$

   - Return to step 1

Complete (for model building) but (almost) never terminates

Proof procedure

Basic ideas

1. Given a formula ϕ of parameter n, consider the two branches:

$$\phi \wedge n = 0 \qquad \phi \wedge n > 0$$

2. The first branch can be handled by usual decision procedures
3. For the second branch:
   - Replace n by $n+1$
   - Simplify the formula:
     - Propositional decomposition rules
     - Equality reasoning rules
     - Expand iterated connectives

$$\bigwedge_{i=0}^{n+1} \phi \rightarrow \bigwedge_{i=0}^{n} \phi \wedge \phi[n+1/i] \qquad \bigvee_{i=0}^{n+1} \phi_i \rightarrow \bigvee_{i=0}^{n} \pi \vee \phi[n+1/i]$$

   - Eliminate all terms indexed by $n+2$
   - Return to step 1

Proof procedure

Termination:

- Index depth is bounded: $0, n, n+1, \cancel{n+2}$
- Number of formulae of bounded index depth is finite

$\Rightarrow$ all infinite branches necessary contain a loop

Proof procedure

Key problem: How to eliminate terms indexed by $n+2$ (preserving sat-equivalence) ?
After decomposition, the formulæ are either:

- Equalities or inequalities or
- Iterated formulæ

$$\bigvee_{i=0}^{n} \phi$$

$$\bigwedge_{i=0}^{n} \phi$$

- $n+2$ does not occur in these formulæ...
  but their truth value still depend on constants a_{n+2} !

Proof procedure

Key problem: How to eliminate terms indexed by $n+2$?
Easy in one particular case:

- $a_{n+2} = c \wedge \phi$ where c is a constant or constant indexed by n, $n+1$

  Replace a_{n+2} by c and delete this equation

Difficult in general:

- $a_{n+2} = \mathit{select}(t, c) \wedge \phi$

  Cannot replace a_{n+2} by $\mathit{select}(t, c)$
  because we cannot increase the depth of the terms !

How to eliminate terms indexed by $n+2$?

The properties of a_{n+2} entail constraints on the other constants:

- $select(f, a_{n+2}) = c \wedge select(g, a_{n+2}) = d \wedge c = d$
  Entails that f and g "agree" at cell a_{n+2}
- $select(f, a_{n+2}) = c \wedge select(g, a_{n+2}) = d \wedge c \neq d$
  Entails that f and g "disagree" at cell a_{n+2}
  Can be contradictory with, e.g., $f =_{\{t\}} g$, if $t \neq a_{n+2}$

How to encode the same information without using a_{n+2} ?

How to eliminate terms indexed by $n+2$?

Solution: enrich the language with additional set symbols

- Denote symbolically the set of terms

$$\{c \mid c \in \Sigma\} \cup \{c_k \mid c \in \Sigma, k \in \mathbb{N}, k \leq n\}$$

- $\theta(n)$: *complement* of this set
  denote "non-named" individuals (up to a bound n on the indices)
- $\theta^c(n)$: $\{c \mid c \in \Sigma\} \cup \{c_k \mid c \in \Sigma, k \in \mathbb{N}, k \leq n\}$
  "named" elements

How to eliminate terms indexed by $n+2$? (2)

- $f =_{\theta(n)} g$ holds if f and g agree at all "non-named" cells
- $f \neq_{\theta(n)^c} g$ holds if f and g disagree at some "non-named" cell
- Provide an abstract description of the behavior of an array "outside" the named constants

How to eliminate terms indexed by $n+2$?

- Use the semantic property (contiguous interpretations):
  - Either a_{n+2} is equal to some constant b_{n+1} or c
  - Or it is distinct from *all* non-indexed constants and from all constants whose index is lower than $n+1$

    i.e. $a_{n+2} \in \theta(n+1)$
- In the first case, a_{n+2} can be eliminated by replacement
- In the second case, additional information on the remaining constants can be derived and expressed using only the special predicate $=_E$ and the set $\theta(n)$

How to eliminate terms indexed by $n+2$?

Add inference rules to derive properties that can be stated using $=_{\theta(n)}$:

$$\frac{select(f, a_{n+2}) = c \wedge select(g, a_{n+2}) = d}{a_{n+2} \notin \theta(n+1) \vee c = d \vee f \neq_{\theta(n+1)^c} g}$$

Remark: the derived formulæ do not refer to $n+2$

How to eliminate terms indexed by $n+2$?

Add new closure rules:

$$\frac{f =_E g, f \neq_{\theta(n)^c} g}{\bot}$$

If $E \cap \theta(n) = \emptyset$

How to eliminate terms indexed by $n+2$?

Add rules to reason on the special sets $\theta(n)$:

- Let T_n denotes the set of constants that are indexed by n or non-indexed
- $t_\alpha \in \theta(\beta) \Leftrightarrow \texttt{false}$ if $\alpha \leq \beta$
- $t_\alpha \in \theta(\beta) \Leftrightarrow (t_\alpha \notin T_{\alpha-1} \vee \exists v \in T_{\alpha-1}\, t_\alpha = v \wedge v \in \theta(\beta))$ otherwise

How to eliminate terms indexed by $n+2$?

Theorem

If a set of formulæ S is irreducible by all expansion rules, then S is satisfiable iff the set of formulæ in S that do not contain $n+2$ is satisfiable.

$$\frac{S \cup S_{n+2}}{S}$$

If $S \cup S_{n+2}$ is irreducible by all the previous rules, all formulæ in S_{n+2} contain $n+2$

Proof procedure

Use various inference rules:

- Usual propositional decomposition rules
- Inductive rules (lazy instantiation of the parameter)
- Equality propagation techniques
- A loop detection rule
- Special rules to reason on the properties of the arrays outside the named constants

Proof procedure

Properties:

- Correct
- Terminating
- Refutationnally complete

A simple application

$i \leftarrow 0$
$j \leftarrow 0$
while $i \leq n$ **do**
  if $P(A[i])$ **then**
    $B[j] \leftarrow A[i]$
    $j \leftarrow j + 1$
  end if
  $i \leftarrow i + 1$
end while

A simple application

The behavior of the program is described by the following schema:

$$
\begin{array}{l}
j_0 = 0 \\
\bigwedge_{i=0}^{n} P(A_i) \not\simeq \texttt{true} \vee B_{i+1} = \mathit{store}(B_i, j_i, A_i) \\
\bigwedge_{i=0}^{n} P(A_i) \not\simeq \texttt{true} \vee j_{i+1} = \mathit{select}(\mathit{succ}, j_i) \\
\bigwedge_{i=0}^{n} P(A_i) = \texttt{true} \vee j_{i+1} = j_i \\
\bigwedge_{i=0}^{n} P(A_i) = \texttt{true} \vee B_{i+1} = B_i
\end{array}
$$

A simple application

Example of properties that can be encoded/checked

The final array satisfies P at cells $j_1, \ldots, j_n$:

$$C = B_{n+1}$$
$$\bigvee_{i=0}^{n} P(\mathit{select}(C, j_i)) \neq \texttt{true}$$

A simple application

Example of properties that can be encoded/checked

All elements occurring in the initial array and satisfying the property P occur in the final array.

$$C = B_{n+1}$$
$$\bigvee_{i=0}^{n}(u = A_i \wedge P(u) = \texttt{true})$$
$$\bigwedge_{i=0}^{n} select(C, j_i) \neq u$$

Summary

Two results:

- Undecidability result for schemata of formulæ in the theory of arrays (even without self-referencing arrays)
- Decidability result for a restricted class of interpretations (contiguous models)

Future work

- Implementation and experimentation
- Refined complexity analysis (double exponential upper bound)
- Identifying automatically formulæ with contiguous models