

Towards SMT Model Checking of Array-based Systems

(Presentation only paper, published in Proc. of IJCAR'08)

S. Ghilardi¹, E. Nicolini², **S. Ranise**³, and D. Zucchelli¹

¹Università degli Studi di Milano

²LORIA & INRIA-Lorraine

³Università di Verona

SMT'08 — July, 7-8 2008



Background

Goal

Lift SMT techniques to **unbounded** model checking of **infinite** state systems

- **Symbolic** model checking
- **First-order formulae** provide finitary representation for sets of **states** and **transitions**
- **Theories** provide support for specifying
 - ▶ the **topology** of the system
 - ▶ the **data structures** manipulated by the system

Systems manipulating arrays

- Examples
 - ▶ Broadcast, mutual-exclusion, cache coherence protocols
 - ▶ Lossy channel systems
 - ▶ Sorting programs
- **Uniform verification problem**: verify correctness **regardless** of the number of elements (processes, data, or integers) in the array
- **Array-based systems**: suitable abstractions of a large class of systems manipulating arrays
- In this talk, focus on **safety** (**invariant**)
- In the paper, results also for a sub-class of **liveness** (**recurrence**)

Array-based system

- [State variable] $a : \text{INDEX} \longrightarrow \text{ELEM}$
- [Topology] $T_I = (\Sigma_I, \mathcal{C}_I)$ is mono-sorted on INDEX st.
 - ▶ $\text{SMT}(T_I)^1$ is **decidable**
 - ▶ T_I is **locally finite** and **closed under substructures**
- [Data structures] $T_E = (\Sigma_E, \mathcal{C}_E)$ is mono-sorted on ELEM st.
 - ▶ $\text{SMT}(T_E)$ is **decidable**
 - ▶ T_E admits **quantifier elimination**

The combined theory $A_I^E = (\Sigma, \mathcal{C})$

- **3 sorts:** INDEX, ELEM, ARRAY
- $\Sigma := \Sigma_I \cup \Sigma_E \cup \{ _[_] : \text{ARRAY} \times \text{INDEX} \longrightarrow \text{ELEM} \}$
- $\mathcal{M} \in \mathcal{C}$ iff $\mathcal{M}|_{\Sigma_I} \in \mathcal{C}_I$ and $\mathcal{M}|_{\Sigma_E} \in \mathcal{C}_E$
 - ▶ $\text{ARRAY}^{\mathcal{M}} := \{ \text{total functions from } \text{INDEX}^{\mathcal{M}} \text{ to } \text{ELEM}^{\mathcal{M}} \}$
 - ▶ $_[_]^{\mathcal{M}} := \text{function application}$

¹ $\text{SMT}(T)$ denotes the Satisfiability of quantifier-free formulae modulo the theory T

Parameterized systems as Array-based systems

- [State variable] $a : \text{INDEX} \longrightarrow \text{ELEM}$
- [Topology] T_I can be
 - ▶ theory of **infinite set** \implies INDEX is a bunch of distinct process identifiers
 - ▶ theory of **total order** \implies INDEX as above with the capability of distinguishing processes on the left/right
 - ▶ most topologies covered **except** rings!
- [Data structures] T_E can be
 - ▶ **enumerated data type** theory \implies finite set $\{q_1, \dots, q_n\}$ of states of each process
 - ▶ **integers** \implies variables of type integers local to each process
 - ▶ many other **data-type** theories covered

SMT and operations on sets of states: **quantifier-free**

- $SMT(A_I^E)$ = combination of dec procs for $SMT(T_I)$ and $SMT(T_E)$ by **unidirectional** Nelson-Oppen schema
- **Key observation**: a conjunction $\psi(i_1, \dots, i_n, a[i_1], \dots, a[i_n], e_1, \dots, e_m)$ of ground literals in A_I^E is **equisatisfiable** to

$$\begin{array}{lll} \psi_I(i_1, \dots, i_n) & \wedge \bigwedge_{k=1}^n a[i_k] = d_k \wedge & \psi_E(d_1, \dots, d_n, e_1, \dots, e_m) \\ \Sigma_I\text{-literals} & \text{func. constraints} & \Sigma_E\text{-literals} \\ SMT(T_I) & i_{k_1} = i_{k_2} \Rightarrow d_{k_1} = d_{k_2} & SMT(T_E) \end{array}$$

- Lift to arbitrary Boolean combinations of atoms in A_I^E by, e.g., (a variant of) **Delayed Theory Combination**

First-order formulae for states and transitions

- $\exists^!$ -formulae: $\exists i_1, \dots, i_n. \phi(i_1, \dots, i_n, a[i_1], \dots, a[i_n])$

Example: mutual exclusion

$$\exists i_1, i_2. i_1 \neq i_2 \wedge a[i_1] = a[i_2] = \text{critical}$$

First-order formulae for states and transitions

- $\exists^!$ -formulae: $\exists i_1, \dots, i_n. \phi(i_1, \dots, i_n, a[i_1], \dots, a[i_n])$

Example: mutual exclusion

$$\exists i_1, i_2. i_1 \neq i_2 \wedge a[i_1] = a[i_2] = \text{critical}$$

First-order formulae for states and transitions

- \exists^l -formulae: $\exists i_1, \dots, i_n. \phi(i_1, \dots, i_n, a[i_1], \dots, a[i_n])$
- \forall^l -formulae: $\forall i_1, \dots, i_n. \phi(i_1, \dots, i_n, a[i_1], \dots, a[i_n])$
Example: initial states (also invariant)

$$\forall i. a[i] = \text{initial}$$

First-order formulae for states and transitions

- \exists^l -formulae: $\exists i_1, \dots, i_n. \phi(i_1, \dots, i_n, a[i_1], \dots, a[i_n])$
 - \forall^l -formulae: $\forall i_1, \dots, i_n. \phi(i_1, \dots, i_n, a[i_1], \dots, a[i_n])$
- Example: **initial states** (also **invariant**)

$$\forall i. a[i] = \text{initial}$$

First-order formulae for states and transitions

- **\exists^l -formulae**: $\exists i_1, \dots, i_n. \phi(i_1, \dots, i_n, a[i_1], \dots, a[i_n])$
- **\forall^l -formulae**: $\forall i_1, \dots, i_n. \phi(i_1, \dots, i_n, a[i_1], \dots, a[i_n])$
- **T-formulae**: $\exists i_1, \dots, i_n. \phi_L(i_1, \dots, i_n, a[i_1], \dots, a[i_n], a'[i_1], \dots, a'[i_n]) \wedge \forall j. \phi_G(i_1, \dots, a[i_1], \dots, a'[i_1], \dots, j, a[j], a'[j])$

Example: a **transition** (from Bakery)

$$\exists i \left(\left(a[i] = \text{critical} \wedge a'[i] = \text{idle} \right) \wedge \forall j \left(\left(j = i \wedge a'[j] = a[j] \right) \vee \left(j \neq i \wedge a'[j] = a[j] \right) \right) \right)$$

Checking invariant (safety) properties

Backward reachability

```
 $i \leftarrow 0$ ;  $BR^0(\tau, K) \leftarrow K$ ;  $K^0 \leftarrow K$   
repeat  
   $K^{i+1} \leftarrow \text{Pre}(\tau, K^i)$   
   $BR^{i+1}(\tau, K) \leftarrow BR^i(\tau, K) \vee K^{i+1}$   
  if  $A_I^E\text{-check}(BR^{i+1}(\tau, K) \wedge I) = \text{sat}$  then return unsafe  
  else  $i \leftarrow i + 1$   
until  $A_I^E\text{-check}(\neg(BR^{i+1}(\tau, K) \rightarrow BR^i(\tau, K))) = \text{unsat}$   
return safe
```

- K : \exists^I -formula (the set of **bad** states)
- I : \forall^I -formula (the set of **initial** states)
- τ : disjunction of T-formulae (**transition**)
- **What about $BR^i(\tau, K)$?** ($i = 0, 1, \dots$)

Checking invariant (safety) properties

Backward reachability

```
 $i \leftarrow 0; BR^0(\tau, K) \leftarrow K; K^0 \leftarrow K$   
repeat  
   $K^{i+1} \leftarrow \text{Pre}(\tau, K^i)$   
   $BR^{i+1}(\tau, K) \leftarrow BR^i(\tau, K) \vee K^{i+1}$   
  if  $A_I^E\text{-check}(BR^{i+1}(\tau, K) \wedge I) = \text{sat}$  then return unsafe  
  else  $i \leftarrow i + 1$   
until  $A_I^E\text{-check}(\neg(BR^{i+1}(\tau, K) \rightarrow BR^i(\tau, K))) = \text{unsat}$   
return safe
```

- What about $BR^i(\tau, K)$? ($i = 0, 1, \dots$)
- **How can we check the A_I^E -satisfiability of quantified formulae?**

Closure under pre-image computation

- Pre-image of an \exists^I -formula K w.r.t. a T-formula τ

$$Pre(\tau, K) := \exists a'(K(a') \wedge \tau(a, a'))$$

Closure under pre-image

$Pre(\tau, K)$ is A_I^E -equiv. to an **effectively computable** \exists^I -formula $K'(a)$

- **Proof** by simple logical manipulations and the **assumption that T_E admits quantifier-elimination**
- **Observe** that $Pre(\bigvee_i \tau_i, K) \Leftrightarrow \bigvee_i Pre(\tau_i, K)$

Back to SMT and operations on sets of states (I)

- Conjunctions of \exists^l - and a \forall^l -formulae:

- ▶ $BR^0(\tau, K) \wedge I \rightsquigarrow K \wedge I$
- ▶ $BR^{i+1}(\tau, K) \wedge I$ because of the hypothesis of closure under pre-image computation
- ▶ $\neg(BR^{i+1}(\tau, K) \rightarrow BR^i(\tau, K)) \rightsquigarrow BR^{i+1}(\tau, K) \wedge \neg BR^i(\tau, K)$
- ▶ To summarize, we need to be able to check the A_I^E -satisfiability of $\exists^l \forall^l$ -formulae

$$\exists i_1, \dots, i_n \forall j_1, \dots, j_m \psi(i_1, \dots, i_n, j_1, \dots, j_m, a[i_1], \dots, a[i_n], a[j_1], \dots, a[j_m])$$

where ψ is quantifier-free

Back to SMT and operations on sets of states (II)

$\exists^l \forall^l$ -formula: $\exists i_1, \dots, i_n \forall j_1, \dots, j_m \psi(i_1, \dots, i_n, j_1, \dots, j_m, a[i_1], \dots, a[i_n], a[j_1], \dots, a[j_m])$

- Variables i_1, \dots, i_n in the existential prefix are Skolem constants
- Variables j_1, \dots, j_m in the universal prefix must be **instantiated!**
- Recall that T_I is assumed to be
 - ▶ locally finite \Rightarrow **only finitely many instances of each j_k must be considered**
 - ▶ closed under substructures \Rightarrow **any finite substructure considered above is a model** (if the case)
- **It is decidable to check the A_I^E -satisfiability of $\exists^l \forall^l$ -formulae**
- **How?** (i) compute representatives of T_I , (ii) instantiate the j_k 's, and then (iii) invoke $SMT(A_I^E)$
 - \Rightarrow interleave (ii) and (iii) for efficiency



Termination of backward reachability

- When the system is safe, algorithm may **diverge**
- To ensure termination, a **standard method** is to define a **well-quasi-ordering** (wqo) on configurations
- In **our framework**, we define a **configuration model-theoretically** as a pair (valuation, **finite-index model**)
- The set $\llbracket K \rrbracket$ of configurations satisfying the $\exists^!$ -formula K is **upward closed** and $\llbracket K_1 \rrbracket \subseteq \llbracket K_2 \rrbracket$ iff $A_I^E \models K_1 \rightarrow K_2$, for $\exists^!$ -formulae K_1, K_2

Upward closure implies termination

T_E if locally finite \Rightarrow backward reachable configurations are finitely generated and upward closed set \Rightarrow backward reachability terminates

UNIVERSITÄT

Conclusions, ongoing, and future work

- **Fully declarative** framework for **unbounded** model checking
- Leveraging state-of-the-art SMT solving
- Ongoing
 - ▶ SMTMC: promising experimental results (see additional slides after this one)
- Future
 - ▶ Efficient implementation of “non-functional” T-formulae
 - ▶ Implement liveness
 - ▶ Dynamic synthesis of invariants while doing backward search
 - ▶ Abstract-check-refine to handle Linear Arithmetic on indexes

Additional slides...

Implementation (I)

- **First attempt: top-down** approach
 - ▶ client-server arch.: generate proof obligation for **arbitrary** SMT solver (using the SMT-LIB format)
 - ▶ T_E : naive **quantifier elimination** for enumerated data-type (used only in pre-image computation)
 - ▶ T_I : naive **quantifier instantiation** for theory of linear orders (to generate only ground proof obligations)
 - ★ actual SMT solvers unsatisfactorily handle quantified formulae **even when simple instances are to be found**
[recall that variables \mapsto Skolem constants]
- **Problem:** it **does not scale up**, problems even with simple mutual exclusion protocols (e.g., Burns, Szymanski)

Implementation (II)

- **Second attempt**: **bottom-up** approach (master project by Thomas Valsecchi of Univ. di Milano)
 - ▶ client-server arch.: generate proof obligation for **one** SMT solver (Yices, supporting incremental sat checks)
 - ▶ **restriction on T-formulae**: roughly only “functional” transition \Rightarrow simpler computation of pre-images, **no** need of **quantifier elimination** for T_E
 - ▶ **lazy** generation of proof obligations, using support for **predication definition** and introducing “**pointers**” to formulae representing states
 - ▶ techniques to pre-process reachability problem and **reduce/avoid quantifier instantiation** for T_I
- **SMTMC**: very encouraging results
Tool and accompanying paper available on request, just send me an email at



silvio.ranise@univr.it

Challenges for SMT solvers

Based on our experience with SMT, we offer here some of the features we would have liked to find available in currently available SMT solvers that would have eased the task of integrating them into our tool

Standard set of minimal **interface** functionalities for **incremental SMT solving** (push/assert, pop/retract, ...)

Built-in **support** for selected **classes of quantified formulae**: avoid to return `unknown` when possible...

[**More difficult**] capability to **retrieve** “simplified” **formulae representing relevant information**, e.g., sets of reachable states

- ▶ useful when performing depth-first search in backward reachability and one wants to use partial information on fix-point checks in other branches...



Acknowledgements

The third author was partially supported by the FP7-ICT-2007-1 Project no. 216471, “AVANTSSAR: Automated Validation of Trust and Security of Service-oriented Architectures” (www.avantssar.eu)

