Finite Models for Safety Verification

Alexei Lisitsa

Department of Computer Science
University of Liverpool,
Liverpool, UK

ENS Cachan, LSV, June 22, 2012

Outline

- Preamble
- Reachability as deducibility
- Verification via countermodel finding
  - Lossy Channel Systems
  - Cache Coherence Protocols
  - **Linear Systems of Automata and Monotonic Abstraction**
  - Regular Model Checking
  - **Regular Tree Model Checking**
  - Safety for general TRS and Tree Automata Completion

Preamble: Fibonacci Words

Fibonacci words:

- $w_o = b$
- $w_1 = a$
- $w_{i+2} = w_{i+1} w_i$
- $b, a, ab, aab, abaab, aababaab, abaabaababaab \ldots$

Preamble: Fibonacci Words

Fibonacci words:

- $w_o = b$
- $w_1 = a$
- $w_{i+2} = w_{i+1} w_i$

- $b, a, ab, aab, abaab, aababaab, abaabaababaab \ldots$
- **Observation:** none of the words contains bb or aaa as the subword

Preamble: Fibonacci Words

Fibonacci words:

- $w_o = b$
- $w_1 = a$
- $w_{i+2} = w_{i+1} w_i$

- $b, a, ab, aab, abaab, aababaab, abaabaababaab \ldots$
- **Observation:** none of the words contains bb or aaa as the subword
- **Question:** How to prove it?

Preamble: Fibonacci Words

Fibonacci words:

- $w_o = b$
- $w_1 = a$
- $w_{i+2} = w_{i+1} w_i$

- $b, a, ab, aab, abaab, aababaab, abaabaababaab \ldots$
- **Observation:** none of the words contains bb or aaa as the subword
- **Question:** How to prove it?
- **Answer:** Let's apply FO logic ...

Preamble: Fibonacci Words

Fibonacci words:

- $w_o = b$
- $w_1 = a$
- $w_{i+2} = w_{i+1} w_i$

- $b, a, ab, aab, abaab, aababaab, abaabaababaab \ldots$
- **Observation:** none of the words contains bb or aaa as the subword
- **Question:** How to prove it?
- **Answer:** Let's apply FO logic ...

Logic encoding

FO theory *FIB*:

- $(x * y) * z = x * (y * z)$
- $R(b, a)$
- $R(x, y) \rightarrow R(y, x * y)$

Proposition

If w is a Fibonacci word then $FIB \vdash \exists x R(t_w, x)$

Here t_w denote a term encoding of w, i.e. $t_{aba} = (a * b) * a$

Corollary

If $FIB \nvdash \exists x \exists z \exists y R(z * b * b * y, x)$ then there is no Fibonacci word with bb as a subword

Finite countermodel

Now to show $FIB \nvdash \exists x \exists z \exists y R(z * b * b * y, x)$ we are looking for

- Finite countermodels for $FIB \rightarrow \exists x \exists z \exists y R(z * b * b * y, x)$, or equivalently, for
- Finite models for $FIB \wedge \neg \exists x \exists z \exists y R(z * b * b * y, x)$

To find a model we apply generic finite model finding procedure, e.g. impemented in Mace4 finite model finder by W.McCune (see demonstration)

Finite countermodels

Now to show $FIB \nvdash \exists x \exists z \exists y R(z * b * b * y, x)$ we are looking for

- Finite countermodels for $FIB \rightarrow \exists x \exists z \exists y R(z * b * b * y, x)$, or equivalently, for
- Finite models for $FIB \wedge \neg \exists x \exists z \exists y R(z * b * b * y, x)$

To find a model we apply generic finite model finding procedure, e.g. impemented in Mace4 finite model finder by W.McCune (see demonstration)

- A model of size 5 is found in 0.05s. The property is proved!

Finite countermodels

Now to show $FIB \nvdash \exists x \exists z \exists y R(z * b * b * y, x)$ we are looking for

- Finite countermodels for $FIB \rightarrow \exists x \exists z \exists y R(z * b * b * y, x)$, or equivalently, for
- Finite models for $FIB \wedge \neg \exists x \exists z \exists y R(z * b * b * y, x)$

To find a model we apply generic finite model finding procedure, e.g. impemented in Mace4 finite model finder by W.McCune (see demonstration)

- A model of size 5 is found in 0.05s. The property is proved!
- To show that $'aaa'$ is not a subword of any Fib. word a model of size 11 can be found in $\approx$ 45s

From Fibonacci words to Safety Verification

In this talk:
the same idea/approach can be applied to (surprisingly) large classes of infinite state and parameterized safety verification problems.

Reachability as deducibility

- Many problems in verification can be naturally formulated in terms of *reachability* within transition systems;
- We propose to use deducibility (or derivability) in first-order predicate logic to model reachability in transition systems of interest;
- Then verification can be treated as theorem (dis)proving in classical predicate logic;
- Many automated tools (provers and model finders) are readily available.

Reachability as deducibility

- Let $\mathcal{S} = \langle S, \rightarrow \rangle$ be a transition system with the set of states S and transition relation $\rightarrow$
- Let $e : s \mapsto \varphi_s$ be encoding of states of $\mathcal{S}$ by formulae of first-order predicate logic, such that
  - the state s' is reachable from s, i.e. $s \rightarrow^* s'$ if and only if $\varphi_{s'}$ is the logical consequence of φ_s, that is $\varphi_s \models \varphi_{s'}$ and $\varphi_s \vdash \varphi_{s'}$.
- Under such assumptions:
  - Establishing reachability $\equiv$ theorem proving
  - Establishing non-reachability $\equiv$ theorem disproving

Verification of safety

- Safety $\equiv$ non-reachability of "bad" states
- Verification of safety properties $\equiv$ theorem disproving
- To disprove $\varphi \models \psi$ it is sufficient to a find a countermodel for $\varphi \to \psi$, or which is the same a model for $\varphi \wedge \neg\psi$
- In general, such a model can be inevitably infinite and the set of satisfiable first-order formulae is not r.e.
- One can not hope for full automation here
- Our proposal: use automated finite model finders/builders

Remarks

- For the verification of safety the weaker assumption on the encoding is sufficient:
  - $s \rightarrow^* s' \Rightarrow \varphi_s \vdash \varphi_{s'}$
- For the verification of parameterized systems general idea of reachability as deducibility should be suitably adjusted
  - depends on particular classes of systems
  - unary or binary predicates modeling reachabiity can be used

Origins

- The idea of using finite model finders for verification is not new (thanks to anonymous referees of FMCAD 2010 conference!)
- It was proposed and developed in the area of verification of security protocols in the following papers (at least):
  - C. Weidenbach Towards an Automatic Analysis of Security Protocols in First-Order Logic, in H. Ganzinger (Ed.): CADE-16, LNAI 1632, pp. 314–328, 1999.
  - Selinger, P.: Models for an adversary-centric protocol logic. Electr. Notes Theor. Comput. Sci. 55(1) (2001);
  - Goubault-Larrecq, J.: Towards producing formally checkable security proofs, automatically. In: Computer Security Foundations (CSF), pp. 224238 (2008)
  - Jan Jurjens and Tjark Weber, Finite Models in FOL-Based Crypto-Protocol Verification. Foundations and Applications of Security Analysis, LNCS 5511, 2009.

What is new then?

- Countermodel finding based verification methods are practically efficient for the verification of various classes of infinite state and parameterized systems:
  - lossy channel systems
  - cache coherence protocols
  - parameterized linear arrays of finite state automata
  - etc.
- Completeness (for lossy channel systems verification)
- Relative completeness wrt to regular model checking (RMC); regular tree model checking (RTMC); tree automata completion techniques
- In many cases no specialized model finding procedures is needed. Generic MACE4 finite model finder by W.McCune has been successfully used to verify above systems

Systems to verify, I

Parameterized linear arrays of automata

Parameterized mutual exclusion protocol

- Taken from the paper Parosh Aziz Abdulla, Giorgio Delzanno, Noomene Ben Henda, Ahmed Rezine. Monotonic Abstraction: on Efficient Verification of Parameterized Systems. *Int. J. Found. Comput. Sci.* 20(5): 779-801 (2009)
- Operates on the parameterized linear array of finite state automata

Protocol specification

The protocol is specified as a parameterized system $\mathcal{ME} = (Q, T)$, where $Q = \{green, black, blue, red\}$ is the set of local states of finite automata, and T consists of the following transitions:

- $\forall_{LR}\{green, black\} : green \rightarrow black$
- $black \rightarrow blue$
- $\exists_{L}\{black, blue, red\} : blue \rightarrow blue$
- $\forall_{L}\{green\} : blue \rightarrow red$
- $red \rightarrow black$
- $black \rightarrow green$

The correctness condition: if the protocol starts with all states being *green* it will never get to a state where there are two or more automata in the *red* state

Translation to the first-order logic,I

- $(x * y) * z = x * (y * z)$
- $e * x = x * e = x$

  ($*$ is a monoid operation and e is a unit of a monoid)

- $G(e)$
- $G(x) \rightarrow G(x * green)$

  (specification of configurations with all green states)

- $GB(e)$
- $GB(x) \rightarrow GB(x * green)$
- $GB(x) \rightarrow GB(x * black)$

  (specification of configurations with all states being green or black)

Translation to the first-order logic,II

- $G(x) \rightarrow R(x)$

  (initial states assumption: "allgreen" configurations are reachable)

- $(R((x * green) * y)\ \&\ GB(x)\ \&\ GB(y)) \rightarrow R((x * black) * y)$
- $R((x * black) * y) \rightarrow R((x * blue) * y)$
- $R((x * blue) * y)\ \&\ (x = (z * black) * w) \rightarrow R((x * blue) * y)$
- $R((x * blue) * y)\ \&\ (x = (z * blue) * w) \rightarrow R((x * blue) * y)$
- $R((x * blue) * y)\ \&\ (x = (z * red) * w) \rightarrow R((x * blue) * y)$
- $R((x * blue) * y)\ \&\ G(x) \rightarrow R((x * red) * y)$
- $R((x * red) * y) \rightarrow R((x * black) * y)$
- $R((x * black) * y) \rightarrow R((x * green) * y)$

  (specification of reachability by one step transitions from T; one formula per transition, except the case with existential condition, where three formulae are used)

Adequacy of encoding and Verification

- If a configuration $\bar{c}$ is reachable in $\mathcal{ME}$ then $\Phi_{\mathcal{P}} \vdash R(t_{\bar{c}})$
- To establish safety property of the protocol (mutual exclusion) it does suffice to show that
  $\Phi_{\mathcal{P}} \nvdash \exists x \exists y \exists z R((((x * red) * y) * red) * z)$.
- Delegate the latter task to the finite model finder MACE4 (see demonstration)

Adequacy of encoding and Verification

- If a configuration $\bar{c}$ is reachable in $\mathcal{ME}$ then $\Phi_{\mathcal{P}} \vdash R(t_{\bar{c}})$
- To establish safety property of the protocol (mutual exclusion) it does suffice to show that
  $\Phi_{\mathcal{P}} \nvdash \exists x \exists y \exists z R((((x * red) * y) * red) * z)$.
- Delegate the latter task to the finite model finder MACE4 (see demonstration)
- It takes approx. 0.01s to find a countermodel and verify the safety property!

Countermodel as Invariant

- Take a configuration $\bar{c}$ of the protocol, consider its term representation $t_{\bar{c}}$
- The following property is an invariant of the system:

$$[t_{\bar{c}}] \in [R]$$

Here $[\ldots]$ denote the interpretation in the (counter)model.

Model and Invariant

The domain D of the model is a four element set $\{0, 1, 2, 3\}$.
Interpretations of constants: $[black] = [blue] = 0$, $[e] = [green] = 1$, $[red] = 2$. Interpretations of unary predicates: $[G] = \{1\}$; $[GB] = \{0, 1\}$; $[R] = \{0, 1, 2\}$.
The interpretation of the binary function $*$ is given by the following table

	0	1	2	3
0	0	0	2	3
1	0	1	2	3
2	2	2	3	3
3	3	3	3	3

Invariant property which holds for any reachable configuration $\bar{c}$:

$$[t_{\bar{c}}] \in [R] = \{0, 1, 2\}$$

Relative completeness

Theorem (2010)

If the safety of parameterized linear system of automata can be demonstrated by monotonic abstraction method then it can be demonstrated by FCM too.

Safety for parameterized linear systems

Problem

Given: *A parameterized system $\mathcal{P} = (Q, T)$, a set $In \subseteq C$ of initial configurations, a set $B \subseteq C$ of bad configurations.*

Question: *Are there any configurations $c \in In$ and $c' \in B$ such that c' is reachable from c in $\mathcal{P}$, i.e. for which $c \rightarrow^*_{\mathcal{P}} c'$ holds?*

A negative answer for the above question means the safety property ("not B") holds for the parameterized system.
Further assumptions:

- $I = q_o^*$ for $q_o \in Q$
- The set B of bad configurations is defined by a finite set of words $F \subseteq Q^*$: $B = \{\bar{c} \mid \exists \bar{w} \in F \ \wedge \ \bar{w} \preceq \bar{c}\}$, where $\bar{w} \preceq \bar{w}'$ denotes that $\bar{w}$ is a (not necessarily contiguous) subword of $\bar{w}'$

Monotonic abstraction

Abdulla et al 2009

Given a parameterized system $\mathcal{P} = (Q, T)$ and the corresponding transition relation $\rightarrow_{\mathcal{P}}$ on the configurations within $\mathcal{P}$.
The monotonic abstraction $\rightarrow^{A}_{\mathcal{P}}$ of $\rightarrow_{\mathcal{P}}$ is as follows. For two configurations $\bar{c}$ and $\bar{c'}$ $\bar{c} \rightarrow^{A}_{\mathcal{P}} \bar{c'}$ holds iff either

- $\bar{c} \rightarrow_{\mathcal{P}} \bar{c'}$ holds, or
- there is a transition $t = \forall_L Jq \rightarrow q'$ in T, $\bar{c} = \bar{c}_l\ q\ \bar{c}_r$ and $\bar{c'} = reduct_J(\bar{c}_l)\ q'\ \bar{c}_r$
- there is a transition $t = \forall_R Jq \rightarrow q'$ in T, $\bar{c} = \bar{c}_l\ q\ \bar{c}_r$ and $\bar{c'} = \bar{c}_l\ q'\ reduct_J(\bar{c}_r)$
- there is a transition $t = \forall_{LR} Jq \rightarrow q'$ in T, $\bar{c} = \bar{c}_l\ q\ \bar{c}_r$ and $\bar{c'} = reduct_J(\bar{c}_l)\ q'\ reduct_J(\bar{c}_r)$

Verification via monotonic abstraction

Symbolic backward reachability algorithm Abdulla et al 2009

- $U_0 = B$
- $U_{i+1} = U_i \cup Pre(U_i)$

where $Pre(X) = \{\bar{c} \mid \exists \bar{c}' \in X \ \wedge \ \bar{c} \rightarrow^A_{\mathcal{P}} \bar{c}'\}$.

- This iterative process is guaranteed to stabilize, i.e $U_{n+1} = U_n$ for some finite n.
- Once the process stabilized the resulting U consists of *all* configurations from which some bad configuration can be reached via $\rightarrow^A_{\mathcal{P}}$.
- Then the check is performed on whether $Init \cap U = \emptyset$. If this condition is satisfied then the safety is established, for no bad configuration can be reached from initial configurations via $\rightarrow^A_{\mathcal{P}}$ and, a fortiori, via $\rightarrow_{\mathcal{P}}$.

Important properties of the fixed-point U

- If the safety holds then $\bar{U}$ (complement of U) is an *invariant* of the system sufficient to prove the safety. Indeed, it subsumes *Init* and is closed under reachability.
- U has a finite set of generators and therefore is a *regular* set. It follows that $\bar{U}$ is a *regular* set too.

Regular Invariants

Theorem (on regular invariants)

Given a parameterized system $\mathcal{P} = (Q, T)$ *and the set of bad configurations* $B = \{\bar{c} \mid \exists \bar{w} \in F \ \wedge \ \bar{w} \preceq \bar{c}\}$. *Then the following two conditions are equivalent:*

(1)There exists a regular set of configurations Inv such that

- $Reach \subseteq Inv$ *where* $Reach = \{y \mid \exists x(x \in Init \wedge x \rightarrow_{\mathcal{P}}^{*} y)\}$
- *Inv is closed under reachability, that is* $x \in Inv \wedge x \rightarrow_{\mathcal{P}} y \Rightarrow y \in Inv$
- $Inv \cap B = \emptyset$

and

(2) There exists a finite model for $\Phi_{\mathcal{P}} \wedge \neg \Psi_F$

Proof.

Uses an algebraic characterization of regular languages in terms of inverse homomorphic images of subsets of finite monoids □

Regular Invariants and Relative Completeness

- If the safety for $\mathcal{P}$ holds and can be shown by the monotonic abstraction method, then

Regular Invariants and Relative Completeness

- If the safety for $\mathcal{P}$ holds and can be shown by the monotonic abstraction method, then
- There exists a regular invariant, which implies

Regular Invariants and Relative Completeness

- If the safety for $\mathcal{P}$ holds and can be shown by the monotonic abstraction method, then
- There exists a regular invariant, which implies
- An existence of a finite countermodel for FO encoding of the problem, which means

Regular Invariants and Relative Completeness

- If the safety for $\mathcal{P}$ holds and can be shown by the monotonic abstraction method, then
- There exists a regular invariant, which implies
- An existence of a finite countermodel for FO encoding of the problem, which means
- Safety can be established by FCM (finite countermodel method) if a complete finite model building procedure is used

Subsets of configurations

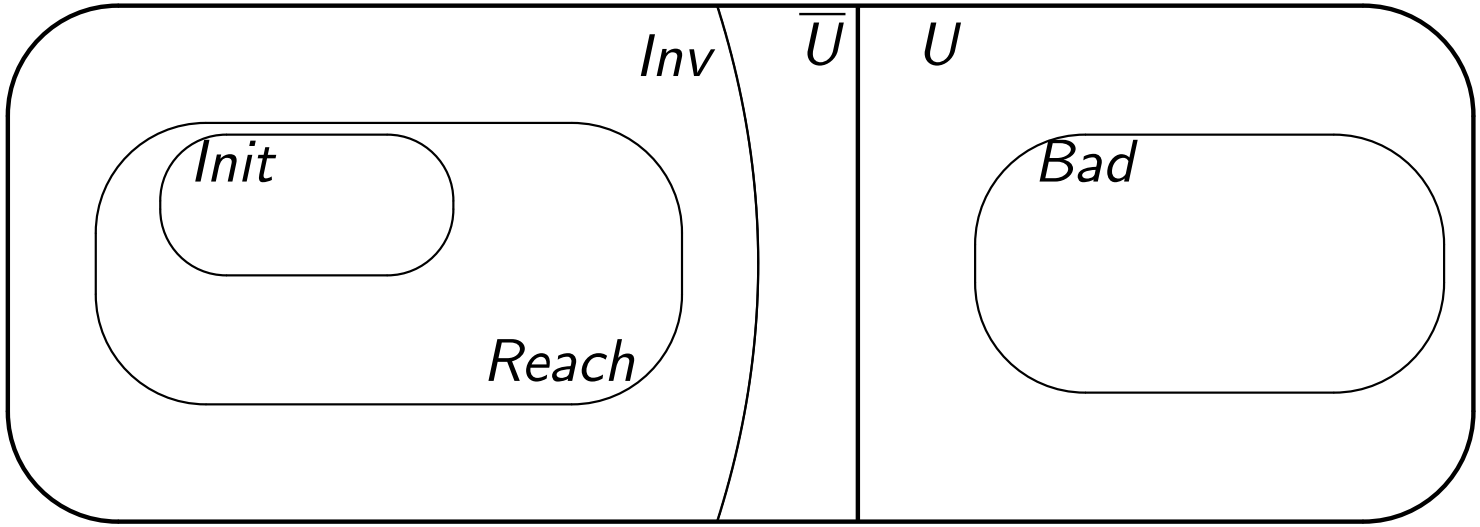

Figure: Subsets of configurations in general position

FCM is stronger than monotonic abstraction

The parameterized system (Q, T) where $Q = \{q_0, q_1, q_2, q_3, q_4\}$ and where T includes the following transition rules

1. $\forall_{LR}\{q_0, q_1, q_4\} : q_0 \rightarrow q_1$
2. $q_1 \rightarrow q_2$
3. $\forall_{L}\{q_0\} : q_2 \rightarrow q_3$
4. $q_3 \rightarrow q_0$
5. $\exists_{LR}\{q_2\} : q_3 \rightarrow q_4$
6. $q_4 \rightarrow q_0$

satisfies mutual exclusion for state q_4, but this fact *can not* be established by the monotonic abstraction method.
Using FCM we have verified mutual exclusion for this system, demonstrating that FCM method is stronger than monotone abstraction. Mace4 has found a finite countermodel of the size 6 in 341s.

Further relative completeness results

Theorem (2010)

If the safety of a linear parameterized system can be demonstrated by regular model checking method then it can be demonstrated by FCM too.

Further relative completeness results

Theorem (2010)

If the safety of a linear parameterized system can be demonstrated by regular model checking method then it can be demonstrated by FCM too.

Theorem (2011)

If the safety of a tree-shape parameterized system can be demonstrated by regular tree model checking method then it can be demonstrated by FCM too.

Further relative completeness results

Theorem (2010)

If the safety of a linear parameterized system can be demonstrated by regular model checking method then it can be demonstrated by FCM too.

Theorem (2011)

If the safety of a tree-shape parameterized system can be demonstrated by regular tree model checking method then it can be demonstrated by FCM too.

Theorem (2011, RTA 2012)

If the safety of a term rewriting system can be demonstrated by tree automata completion technique then it can be demonstrated by FCM too.

Why does it work?

In all cases the proofs of relative completeness results rely upon existence of regular invariants, that is regular sets (of words or trees) subsuming all reachable states and disjoint with all unsafe states.

Parameterized tree-like systems

Verification of parameterized tree-like systems

- Verifcation of safety (a system never goes into a failure state)

Verification of parameterized tree-like systems

- Verifcation of safety (a system never goes into a failure state)
- Tree-like systems: protocols and algorithms operating in the networks with tree topology

Verification of parameterized tree-like systems

- Verifcation of safety (a system never goes into a failure state)
- Tree-like systems: protocols and algorithms operating in the networks with tree topology
- Systems are specified by tree transducers, initial and unsafe configurations are specified by tree automata

Verification of parameterized tree-like systems

- Verifcation of safety (a system never goes into a failure state)
- Tree-like systems: protocols and algorithms operating in the networks with tree topology
- Systems are specified by tree transducers, initial and unsafe configurations are specified by tree automata
- Parameterized verification: safety holds for all sizes of networks

Verification of parameterized tree-like systems

- Verifcation of safety (a system never goes into a failure state)
- Tree-like systems: protocols and algorithms operating in the networks with tree topology
- Systems are specified by tree transducers, initial and unsafe configurations are specified by tree automata
- Parameterized verification: safety holds for all sizes of networks
- **Regular Tree Model Checking (RTMC)** is a traditional approach

Verification of parameterized tree-like systems

- Verifcation of safety (a system never goes into a failure state)
- Tree-like systems: protocols and algorithms operating in the networks with tree topology
- Systems are specified by tree transducers, initial and unsafe configurations are specified by tree automata
- Parameterized verification: safety holds for all sizes of networks
- **Regular Tree Model Checking (RTMC)** is a traditional approach
- We show that FCM method using disproving in first-order logic can be used instead

Tree Automata and Transducers

Definition

A tree automaton over a ranked alphabet Σ is a triple $A = (Q, F, \delta)$, where Q is a finite set of states, $F \subseteq Q$ is a set of final states, and δ is a transition relation, represented by a finite set of rules of the form $(q_1, \ldots, q_p) \rightarrow^f q$, where $f \in \Sigma_p$ and $q_1, \ldots q_p, q \in Q$. A tree automaton over an alphabet Σ^2 is called *tree transducer*.

- The language $L(A)$ of trees accepted by an automaton A is defined as usual.
- For a tree transducer D over an alphabet Σ^2 one-step transition relation $R_D \subseteq T(\Sigma) \times T(\Sigma)$ is defined as $R^D = \{(T, T') \mid T \times T' \text{ is accepted by } D\}$.
- R^i denotes the ith power of R i.e. i compositions of R. $R^* = \cup_{i \geq 0} R^i$.
- For any $L \subseteq T(\Sigma)$ and $R \subseteq T(\Sigma) \times T(\Sigma)$ we denote by $L \star R$ the set $\{y \mid \exists x (x, y) \in L \times T(\Sigma) \cap R\}$.

Regular Tree Model Checking

- Regular Tree Model Checking (RTMC) is a general method for the verification of parameterized systems that have tree topology [Abdulla et al, 2002]
- Regular Tree Model Checking deals with the following basic verification task.

Problem

Given two tree automata A_I and A_U over an alphabet Σ and a tree transducer D over Σ^2. Does $(L(A_I) \star R_D^*) \cap L(A_U) = \emptyset$ hold?

- The verification in RTMC proceeds by producing a tree transducer TR approximating R_D^* from above, that is $R_D^* \subseteq L(TR)$, and showing the emptiness of the set $(L(A_I) \star L(TR)) \cap L(A_U)$

From RTMC to First-Order Logic

- We show that basic RTMC verification problem can be reduced to a purely logical problem of disproving a first-order formula
- Assume we are given an instance of the basic verification problem (over ranking alphabet Σ), that is
  - a tree automaton $A_I = (Q_I, F_I, \delta_I)$ accepting a regular set of initial states;
  - a tree automaton $A_U = (Q_U, F_U, \delta_U)$ accepting a regular set of unsafe states;
  - a tree transducer $D = (Q_D, F_D, \delta_D)$ representation one-step transition relation R_D.
- Now define a set formulae of first-order predicate logic as follows.

Vocabulary

- constants for all elements of $Q_I \sqcup Q_U \sqcup Q_D \sqcup \Sigma_0$;
- unary predicate symbols $Init^{(1)}$, $Unsafe^{(1)}$;
- binary predicate symbols $Init^2$, $Unsafe^2$, R;
- a ternary predicate symbol T;
- a p-ary functional symbol f_θ for every $\theta \in \Sigma_p$

Term translation

Given any tree τ from $T(\Sigma)$ define its term translation t_τ by induction:

- $t_\tau = c$ for a tree τ with one node labeled by $c \in \Sigma_0$;
- $t_\tau = f_\theta(t_{\tau_1}, \ldots, t_{\tau_p})$ for a tree τ with the root labeled by $\theta \in \Sigma_p$ and children $\tau_1, \ldots \tau_p$.

Frist-Order translation of RTMC problem

- Now we are ready to define a translation of RTMC problem, given by tree automata A_I, A_U and a tree transducer D to a set Φ of first-order formulae.
- Φ consists of the following formulae...

Translation of the automaton A_I

- $Init^{(2)}(a, q)$ for every $a \in \Sigma_0$, $q \in Q_I$ and $\rightarrow^a q$ in δ_I;
- $Init^{(2)}(x_1, q_1) \wedge \ldots \wedge Init^{(2)}(x_p, q_p) \rightarrow Init^{(2)}(f_\theta(x_1, \ldots, x_p), q)$ for every $(q_1, \ldots, q_p) \rightarrow^\theta q$ in δ_I;
- $\vee_{q \in F_I} Init^{(2)}(x, q) \rightarrow Init^{(1)}(x)$;

Translation of the automaton A_U

- $Unsafe^{(2)}(a, q)$ for every $a \in \Sigma_0$, $q \in Q_I$ and $\rightarrow^a q$ in δ_I;
- $Unsafe^{(2)}(x_1, q_1) \wedge \ldots \wedge Unsafe^{(2)}(x_p, q_p) \rightarrow Unsafe^{(2)}(f_\theta(x_1, \ldots, x_p), q)$ for every $(q_1, \ldots, q_p) \rightarrow^\theta q$ in δ_I;
- $\vee_{q \in F_I} Unsafe^{(2)}(x, q) \rightarrow Unsafe^{(1)}(x)$;

Translation of the transducer D

- $T(a, b, q)$ for every $\rightarrow^{(a,b)} q$ in δ_D;
- $T(x_1, y_1, q_1) \wedge \ldots \wedge T(x_p, y_p, q_p) \rightarrow T(f_{\theta_1}(x_1, \ldots, x_p), f_{\theta_2}(y_1, \ldots, y_p), q)$ for every $(q_1, \ldots, q_p) \rightarrow^{\theta_1,\theta_2} q$ in δ_D;
- $\vee_{q \in F_D} T(x, y, q) \rightarrow R(x, y)$;

plus reflexive-transitive closure axioms for R:

- $R(x, x)$;
- $R(x, y) \wedge R(y, z) \rightarrow R(x, z)$.

Adequacy of translation

Proposition

If $\tau \in L(A_I)$ *then* $\Phi \vdash \mathit{Init}^{(1)}(t_\tau)$
If $\tau \in L(A_U)$ *then* $\Phi \vdash \mathit{Unsafe}^{(1)}(t_\tau)$
If $\tau \in L(A_I) \star R_D^*$ *then* $\Phi \vdash \exists x\ \mathit{Init}^{(1)}(x) \wedge R(x, t_\tau)$

Corollary

(correctness of the verification method)
If $\Phi \nvdash \exists x \exists y (\mathit{Init}^{(1)}(x) \wedge R(x, y) \wedge \mathit{Unsafe}^{(1)}(y)$ *then*
$(L(A_I) \star R_D^*) \cap L(A_U) = \emptyset$

FCM verification method for RTMC

- In order to prove safety, that is $(L(A_I) \star R_D^*) \cap L(A_U) = \emptyset$ it is sufficient to demonstrate
  $\Phi \nvdash \exists x \exists y (Init^{(1)}(x) \wedge R(x,y) \wedge Unsafe^{(1)}(y)$
- In the FCM method we delegate this task to the generic finite model finding procedure, which searches for the finite countermodels for
  $\Phi \rightarrow \exists x \exists y (Init^{(1)}(x) \wedge R(x,y) \wedge Unsafe^{(1)}(y)$.
- If a countermodel is found the safety is established!

FCM vs RTMC

Theorem (relative completeness of FCM)

Given an instance of the basic verification problem for RTMC, that is two tree automata A_I and A_U over an alphabet Σ and a tree transducer $D = (Q_D, F_D, \delta_D)$ over Σ^2. If there exists a regular tree language $\mathcal{R}$ such that $(L(A_I) \star R_D^*) \subseteq \mathcal{R}$ and $\mathcal{R} \cap L(A_U) = \emptyset$ then there is a finite countermodel for
$\Phi \rightarrow \exists x \exists y (Init^{(1)}(x) \wedge R(x, y) \wedge Unsafe^{(1)}(y))$

It follows then that FCM is at least as powerful in establishing safety as RTMC, provided a complete finite model finding procedure is used.

The case study

Parameterized Two-Way Token protocol:

- The system consists of finite-state processes connected to form a binary tree structure
- Each process stores a single bit which represents the fact that the process has a token
- During operation of the protocol the token can be passed up or down the tree.
- The correctness condition is that no two or more tokens ever appear (if started with one token)
- In parameterized verification we would like to establish correctness for all possible sizes of trees.

(see Demo)

Lossy channel systems

Lossy Channel Systems (LCS)

- Essentially finite state machines equipped with the channels
  - during transitions the messages can be sent into the channels
  - during transitions the messages can be read off the channels
  - the messages can be lost
  - but the order of the messages can not change
- Can be used to specify and verify the protocols on behavioral level
- Due to unbounded channels these are, generally infinite state systems
- Verification of safety is decidable Abdulla, Jonsson, 93

Lossy channel systems by FCM

Theorem

Parallel composition of

- *complete theorem proving and*
- *complete finite model building*

for first-order predicate logic provides with a decision procedure for safety properties of lossy channel systems.

ATVA 2010

What else?

- Parameterized cache coherence protocols specified in terms of Extended FSM ATVA 2010

What else?

- Parameterized cache coherence protocols specified in terms of Extended FSM ATVA 2010
- Safety verification for general term rewriting RTA 2012

What else?

- Parameterized cache coherence protocols specified in terms of Extended FSM ATVA 2010
- Safety verification for general term rewriting RTA 2012

Safety for TRS

- Consider the TRS $\mathcal{R} = \{f(x) \to f(s(s(x)))\}$ and assume that we want to prove that $f(a) \not\to^* f(s(a))$ (automatically)

Safety for TRS

- Consider the TRS $\mathcal{R} = \{f(x) \to f(s(s(x)))\}$ and assume that we want to prove that $f(a) \not\to^* f(s(a))$ (automatically)
- Traditional approach (Genet & Rusu, 2010): use tree automata completion + finite state abstraction, expressed equationally (and added manually)

Safety for TRS

- Consider the TRS $\mathcal{R} = \{f(x) \to f(s(s(x)))\}$ and assume that we want to prove that $f(a) \not\to^* f(s(a))$ (automatically)
- Traditional approach (Genet & Rusu, 2010): use tree automata completion + finite state abstraction, expressed equationally (and added manually)
- Alternative approach we advocate in this talk: translate the original question into a logical problem of disproving a first-order formlua (in classical predicate logic)

How it works

- Let $\Phi = \{R(f(x), f(s(s(x)))),$
  $R(x,y) \wedge R(y,z) \rightarrow R(x,z),$
  $R(x,y) \rightarrow R(f(x), f(y))\}$
- Suppose to the contrary $f(a) \rightarrow^* f(s(a))$
- Then $\Phi \vdash R(f(a), f(s(a)))$ holds in classial first-order logic (a derivation would follow rewriting)
- But one can easily find a finite countermodel for $\Phi \rightarrow R(f(a), f(s(a))$ automatically using a generic finite model finding procedure (e.g. implemented in Mace4)
- So, the original question is resolved positively

Experimental results

Protocol	Time
ABP*	0.93s
MSI	0.01s
MESI	0.03s
MOESI	0.05s
Firefly	0.03s
Synapse N+1	0.01s
Illinois	0.03s
Berkeley	0.03s
Dragon	0.05s
Futurebus+	1.14s
Bakery	0.01s
MutEx	0.01s

* ABP = Alternating Bit Protocol

Practical results.I

Monotonic abstraction and RMC problems.

Protocol	Time
Token passing (non-optimized)	0.12s
Token passing (optimized)	0.01s
Mutual exclusion I	0.03s
Mutual exclusion II	341s
Bakery	0.03s
Paterson$^-$	0.77s

FCM vs RTMC and Monotonic Abstraction

Protocol	Time	Time reported for RTMC*
Token	0.02s	0.06s
Two-way Token	0.03s	0.09s

* the system configuration was *Intel Centrino 1.6GHZ with 768MB of RAM*

Protocol	Time	Time reported*
Token	0.02s	1s
Two-way Token	0.03s	1s
Percolate	0.02s	1s
Leader Election	0.03s	1s
Tree-arbiter	0.02s	37s
IEEE 1394	0.04s	1h15m25s

* the system configuration was *dual Opteron 2.8 GHZ with 8 GB of RAM*

Conclusion

- We presented FCM method for verification of parameterized systems

Conclusion

- We presented FCM method for verification of parameterized systems
- FCM is simple

Conclusion

- We presented FCM method for verification of parameterized systems
- FCM is simple
- FCM is at least as powerful as methods based on monotonic abstractio, RMC, RTMC, tree automata completion techniques in establishing safety

Conclusion

- We presented FCM method for verification of parameterized systems
- FCM is simple
- FCM is at least as powerful as methods based on monotonic abstractio, RMC, RTMC, tree automata completion techniques in establishing safety
- FCM is efficient in practice

Conclusion

- We presented FCM method for verification of parameterized systems
- FCM is simple
- FCM is at least as powerful as methods based on monotonic abstractio, RMC, RTMC, tree automata completion techniques in establishing safety
- FCM is efficient in practice

References

- Lossy Channel Systems ATVA 2010
- Cache Coherence Protocols ATVA 2010
- Linear Systems of Automata and Monotone Abstraction CoRR abs/1011.0447: (2010)
- Regular Model Checking CoRR abs/1011.0447: (2010)
- Regular Tree Model Checking CoRR abs/1107.5142:(2011), TTATT 2012 Workshop, Nagoya, RTA'12
- Safety for general TRS and Tree Automata Completion RTA 2012

Thank you!