Electronic Communications of the EASST
No \volume defined!

Reachability as deducibility, finite countermodels and verification

Alexei Lisitsa

6 pages

No *ed(s) defined!
Managing Editors: Tiziana Margaria, Julia Padberg, Gabriele Taentzer
ECEASST Home Page: http://www.easst.org/eceasst/ ISSN 1863-2122

Reachability as deducibility, finite countermodels and verification

Alexei Lisitsa[1]

[1] A.Lisitsa@liverpool.ac.uk,
Department of Computer Science, the University of Liverpool,
Ashton Building,Ashton Street,
Liverpool, L69 7ZF, U.K.

Abstract: We propose a simple but powerful approach to the verification of parameterised systems. The approach is based on modelling the reachability between parameterized states as deducibility between suitable encodings of states by formulae of first-order predicate logic. To establish a safety property, that is non-reachability of unsafe states, the finite model finder is used to find a finite countermodel, the witness for non-deducibility.

Keywords: automated verification, infinite state systems, parameterised systems, first-order logic, finite model finders, theorem provers

1 Main idea of the method

Let $\mathscr{S} = \langle S, \rightarrow \rangle$ be a transition system with the set of states S and transition relation $\rightarrow$. Denote by $\rightarrow^*$ the transitive closure of $\rightarrow$. Consider encoding $e : s \mapsto \varphi_s$ of states of $\mathscr{S}$ by formulae of first-order predicate logic, satisfying the folowing property. 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 $\varphi_s \vdash \varphi_{s'}$. Here we assume standard definitions of semantical consequence $\models$ and deducibility $\vdash$ (in a complete deductive system) for first-order predicate logic. Under such assumptions one can translate reachability questions for S to the classical questions in logic. Establishing reachability amounts to theorem proving, while deciding non-reachability becomes theorem disproving. It is clear that due to undecidability of first-order logic such an approach can not be universal. However one may hope that much developed automated theorem provers and model finders for first-order logic can be used for automated decision of (non-)reachability problems.

In this paper we will focus on applications of these ideas to the automated verification of *safety* properties of *infite state* and *parameterised* systems. Restriction to the safety properties, i.e. non-reachability of *unsafe* states means we will be mainly dealing with automated disproving. To disprove $\varphi_s \models \varphi_{s'}$ is is sufficient to find a countermodel for $\varphi_s \rightarrow \varphi_{s'}$, or, which is the same, the model for $\varphi_s \wedge \neg \varphi_{s'}$. In general, in first-order logic such a model may be inevitably infinite. Furthermore, the set of satisfiable first-order formulae in not recursively enumerable, so one can not hope for complete automation here. As a partial solution we propose to use automated *finite* model finders/builders [2]. Here we present preliminary results realated to instantiations of these ideas to the verification of lossy channel systems [1] and to the verification of parameterised cache coherence protocols [3].

1.1 Verification of Lossy Channel Systems

Lossy Channel Systems are essentially finite-state automata augmented witha finite amount of unbounded but lossy FIFO channels (queues). The messages sent via lossy channels may be lost in the transition. In [1] we find the definition of a Lossy Channel System L as a tuple $\langle S, s_0, A, C, M, \delta \rangle$, where S is a finite set of *control states*, $s_0 \in S$ is an *initial control state*,A is a finite set of *actions*, C is a finite set of channels, M is a finite set of messages, δ is a finite set of transitions. Starting in the intial control state s_0 the system L may execute non-deterministically any applicable transition from δ, which involves switching the control states and either, writing a message m into some channel c, or reading a message m from some channel c, or just executing an action a. Additionally, at every step any message from any channel may be lost. The sequence of actions $\sigma \in A^*$ executed up to some step is called a trace of L. The set of all traces of L is denoted by $Traces(L)$. A *global state* of L is a pair $\langle s, w \rangle$, where $s \in S$ and $w : C \to M^*$ is a function assigning to each channel a finite sequence of messages (content of the channel).

The general form of the *safety* verification problem for lossy channel systems we address here is as follows.

Given: A lossy channel system $L = \langle S, s_0, A, C, M, \delta \rangle$ and a regular set $\Sigma \subseteq A^*$

Question: Does $Traces(L) \subseteq \Sigma$ hold?

We assume that Σ is effectively given by a deterministic[1] finite automaton $M_{\bar{\Sigma}}$ which accepts the *complement* of Σ. Following the standard approach [1] we first reformulate equivalently the above question as a question on *reachability*:

Is it true that in $L \times M_{\bar{\Sigma}}$ no global state of the form $\langle \langle s, t \rangle, w \rangle$ with $t \in F$ is reachable?

Here $L \times M_{\bar{\Sigma}}$ is a lossy channel system, which is a *product* of L and $M_{\bar{\Sigma}}$ synchronized over actions from A, $\langle \langle s, t \rangle, w \rangle$ is a global state of $L \times M_{\bar{\Sigma}}$ and F is the set of accepting states of $M_{\bar{\Sigma}}$.

1.2 Verification via countermodel finding

In this subsection we show how to apply reachability as deducibility concept and finite countermodel finding for deciding the above reachability problem.

First, we define a translation of the product system $L \times M_{\bar{\Sigma}}$ into a formula of the first-order predicate logic $\Phi_{L \times M_{\bar{\Sigma}}}$ as follows. The vocabulary of $\Phi_{L \times M}$ consists of constant symbols to denote the control states of L and $M_{\bar{\Sigma}}$, the messages of L; the constant symbol e to denote the empty sequence of messages; one binary associative symbol $*$ to denote concatenation and to encode sequences of messages; for every action a a unary functional symbol f_a; and one relational symbol R of arity $n+2$, where n is a number of channels in L. The global state γ of $L \times M_{\bar{\Sigma}}$ is encoded then naturally as a $n+2$-tuple of terms $\bar{t_\gamma}$, the first two terms are to represent the control states of L and $M_{\bar{\Sigma}}$, and the remaining n terms are to encode the content of the channels.

[1] the restriction to deterministic automata leads to more concise translations of *lcs*'s to first-order formulae, but is not very essential in the proposed approach

The intended meaning of the atomic formula $R(\overline{t_\gamma})$ is “the global state γ is reachable”, and the whole formula $\Phi_{L\times M_{\overline{\Sigma}}}$ axiomatizes the reachability in $L\times M_{\overline{\Sigma}}$:

Theorem 1 *The global state γ is reachable in $L\times M_{\overline{\Sigma}}$ if and only if $\Phi_{L\times M_{\overline{\Sigma}}}\vdash R(\overline{t_\gamma})$.*

Let F be a set of acceptable states of $M_{\overline{\Sigma}}$ and d_s be a constant denoting any $s\in F$. Let then B be a first-order sentence $\vee_{s\in F}\exists y\exists\bar{x}R(y,d_s,\bar{x})$.

Theorem 2 *Either $\Phi_{L\times M_{\overline{\Sigma}}}\vdash B$, or there is a* finite *model for $\Phi_{L\times M_{\overline{\Sigma}}}\wedge\neg B$.*

The proof of the Theorem 2 uses the completeness of the symbolic reachability algorithm for the lossy channel systems from [1], in particular the *finite* characterization of the set of global states backwards reachable from any upwards closed[2] set of global states. Based on both theorems, the following is a complete decision procedure for the safety problem for lossy channel systems.

Run in parallel the complete theorem prover for the first-order logic with the input $\Phi_{L\times M_{\overline{\Sigma}}}\to B$ and the complete finite model finder for the first-order logic with the input $\Phi_{L\times M_{\overline{\Sigma}}}\wedge\neg B$ until exactly one successfully returns.

In practical experiments with that procedure we have used a combination of the prover Prover9 and the model finder Mace4 [4], which provides with the convenient unified interface. See the Appendix for the report on the verification of Alternating Bit Protocol modelled as a *lcs*.

1.3 Verification of parameterized cache coherence protocols

Another class of the systems to which verification via finite countermodel finding approach has been applied is parameterized cache coherence protocols [3], modelled by Extended Finite State Machines(EFSM). The states of EFSM are non-negative integer vectors and transitions are affine transformations with affine pre-conditions. The safety is expressed as the non-reachability of global states which belong to some upwards closed sets. We define a translation of the EFSM model M and the correctness conditions C to the first-order formulae ϕ_M and $\psi_C\equiv\vee_i\exists\overline{x_i}\psi_i(\overline{t_i})$, respectively, such that the following proposition holds.

Proposition 1 *If there is a finite model for $\phi_M\wedge\neg\psi_C$ then M satisfies C*

Using the finite model finder Mace4 we have verified all cache coherence protocols from [3]. One example, the verification of Futurbus protocol, is given in the Appendix B.

Bibliography

[1] Parosh Aziz Abdulla, Jonsson B. Verifying programs with unreliable channels. *Information and Computation*, 127(2):91-101, June 15, 1996.

[2] R. Caferra, A. Leitsch, N. Peltier, *Automated Model Building*, Applied Logic Series, 31, Kluwer, 2004.

[3] G. Delzanno. Constraint-based Verification of Parametrized Cache Coherence Protocols. *Formal Methods in System Design*, 23(3):257–301, 2003.

[4] W. McCune Prover9 and Mace4 `http://www.cs.unm.edu/~mccune/mace4/`

[2]with respect to an embeddability pre-order

Appendix A. Verification of Alternating Bit Protocol

The first-order translation Φ of the specification of ABP given in terms of lossy channel systems in [1].

```
% Protocol ABP, first-order translation of the specification given in Abdulla and Jonsson paper,
% Prover9/MACE syntax
 % The positions  of arguments: R(Sender,Receiver,Message, Ack, Automaton)

R(1,1,e,e,1).

R(1,x,y,z,w) -> R(2,x,y,z,s(w)).
R(2,x,y,z,w) -> R(2,x,0 * y,z,w).
R2,x,y,z * 1,w) -> R(2,x,y,z,w).
R(2,x,y,z * 0,w) -> R(3,x,y,z,w).
R(3,x,y,z,w) -> R(4,x,y,z,s(w)).
R(4,x,y,z,w) -> R(4,x,1 * y,z,w).
R(4,x,y,z * 0,w) -> R(4,x,y,z,w).
R(4,x,y,z * 1,w) -> R(1,x,y,z,w).

R(x,1,y,z,w) -> R(x,1,y,1 * z,w).
R(x,1,y * 1,z,w) -> R(x,1,y,z,w).
R(x,1,y * 0,z,w) -> R(x,2,y,z,w).
R(x,2,y,z,w) -> R(x,3,y,z,r(w)).
R(x,3,y,z,w) -> R(x,3,y,0 * z,w).
R(x,3,y * 0,z,w) -> R(x,3,y,z,w).
R(x,3,y,z,w) -> R(x,4,1 * y,z,w).
R(x,4,y,z,w) -> R(x,1,y,z,r(w)).

R(x,y,(z1 * z2) * z3,v,w) -> R(x,y,z1 * z3,v,w).
R(x,y,z,(v1 * v2) * v3,w) -> R(x,y,z,v1 * v3,w).

(x * y) * z = x * ( y * z).

s(1) = 2.
s(2) = 3.
r(1) = 3.
r(2) = 1.
```

The first-order translation Ψ of the the (negation of) correctness condition of ABP.

```
% Prover9/MACE syntax

exists x exists y exists z  exists w R(x,y,z,w,3)
```

The finite model for $\Phi \wedge \neg\Psi$ was found in 0.97 seconds by Mace4 running on the laptop of average specification.

Appendix B. Verification of Futurbus parameterized cache coherence protocol

The first-order translation Φ of EFSM model of the Futurebus+ protocol.

```
% Protocol FutureBus, counting abstraction, first-order translation.
%The syntax of Prover9/MACE. R(..) stands for reachable global state.

plus(0,y) = y.
plus(i(x),y) = i(plus(x,y)).

%-(i(x) = x).

R(i(x1),x2,x3,x4,x5,0,x7,x8,x9) -> R(x1,0,0,0,i(x5),0,plus(x4,x7),x8,plus(x2,plus(x3,x9))).

R(x1,x2,x3,x4,x5,x6,i(x7),x8,x9) ->
R(x1,plus(i(0),plus(x2,x5)),x3,x4,0,x6,x7,x8,x9).

R(x1,x2,x3,x4,x5,x6,x7,x8,i(x9)) ->
R(x1,i(plus(x2,plus(x5,x9))),x3,x4,0,x6,x7,x8,0).

R(x1,x2,x3,x4,i(i(x5)),x6,0,x8,0) ->
R(x1,i(i(plus(x5,x2))),x3,x4,0,x6,0,x8,0).

R(x1,x2,x3,x4,i(0),x6,0,x8,0) ->
R(x1,x2,i(x3),x4,0,x6,0,x8,0).

R(i(x1),x2,x3,x4,x5,0,x7,x8,x9) ->
R(plus(x1,plus(x3,plus(x2,plus(x9,plus(x5,x7))))),0,0,0,0,i(0),0,plus(x4,x8),0).


R(x1,x2,x3,x4,x5,x6,x7,i(x8),x9) ->
R(i(x1),x2,x3,plus(x6,x4),x5,0,x7,x8,x9).

R(x1,x2,x3,x4,x5,x6,x7,0,x9) ->
R(x1,x2,x3,plus(x6,x4),x5,0,x7,0,x9).

R(x1,x2,i(x3),x4,x5,x6,x7,x8,x9) ->
R(x1,x2,x3,i(x4),x5,x6,x7,x8,x9).

R(x1,i(x2),x3,x4,x5,x6,x7,x8,x9) ->
R(plus(x2,x1),0,x3,i(x4),x5,x6,x7,x8,x9).

R(i(x),0,0,0,0,0,0,0,0).
```

The first-order translation Ψ of the (negation of) correctness condition for the Futurebus+ protocol.

```
exists x1 exists x2 exists x3 exists x4 exists x5 exists x6
exists x7 exists x8 exists x9 R(x1,x2,i(i(x3)),x4,x5,x6,x7,x8,x9) |

exists x1 exists x2 exists x3 exists x4 exists x5 exists x6
exists x7 exists x8 exists x9 R(x1,x2,i(x3),i(x4),x5,x6,x7,x8,x9) |

exists x1 exists x2 exists x3 exists x4 exists x5 exists x6
exists x7 exists x8 exists x9 R(x1,x2,x3,i(i(x4)),x5,x6,x7,x8,x9) |

exists x1 exists x2 exists x3 exists x4 exists x5 exists x6
exists x7 exists x8 exists x9 R(x1,i(x2),i(x3),x4,x5,x6,x7,x8,x9) |

exists x1 exists x2 exists x3 exists x4 exists x5 exists x6
exists x7 exists x8 exists x9 R(x1,i(x2),x3,i(x4),x5,x6,x7,x8,x9).
```

The model for $\Phi \wedge \neg\Psi$ was found in 1.14 seconds by Mace4 running on the laptop of average specification.