Finite Countermodel Finding for Infinite State and Parametrized Verification

Alexei Lisitsa

Department of Computer Science
University of Liverpool,
Liverpool, UK

Summer School, CSSR 2019, Novosibirsk, June 29, 2019

Overview

- Preamble: Fibonacci Words, MIU system and MU puzzle
- Reachability as deducibility
- Verification via countermodel finding
  - **Cache Coherence Protocols**
  - **Linear Systems of Automata and Monotonic Abstraction**
  - Regular Model Checking
  - Regular Tree Model Checking
  - Lossy Channel Systems
  - Safety for general TRS and Tree Automata Completion
  - **PhysicalSafetyandSecurity***
  - **Limitations and Challenges**

Preamble: Fibonacci Words

Fibonacci words:

- $w_o = b$
- $w_1 = a$
- $w_{i+2} = w_{i+1} w_i$
- $b, a, ab, aba, abaab, abaababa, abaababaabaab \ldots$

Preamble: Fibonacci Words

Fibonacci words:

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

- $b, a, ab, aba, abaab, abaababa, abaababaabaab \ldots$
- **Observation:** none of the words contains bb or aaa as the subword

Preamble: Fibonacci Words

Fibonacci words:

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

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

Preamble: Fibonacci Words

Fibonacci words:

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

- $b, a, ab, aba, abaab, abaababa, abaababaabaab \ldots$
- **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:

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

- $b, a, ab, aba, abaab, abaababa, abaababaabaab \ldots$
- **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 $\approx$ 43s (2019)

From Fibonacci words to Safety Verification

In this lecture:
The same idea/approach can be applied to (surprisingly) large classes of infinite state and parameterized safety verification problems. Furthermore it is competitive with many alternative approaches.

MIU system

Alphabet: M, I and U
Axiom: MI
Derivation rules:

I. If xI is a theorem, so is xIU.

II. If Mx is theorem, so is Mxx.

III. In any theorem III can be replaced by U.

IV. UU can be dropped from any theorem.

MU puzzle

Is MU a theorem of MIU system?

Douglas Hofstadter, Goedel, Escher, Bach: An eternal Golden Braid, 1979

MU puzzle

- **Answer:** Negative, that is $MU \notin L_{MIU}$
- **Condition, 1 (GEB,79):** "the number of I symbols in any string in L_{MIU} cannot be multiple of three"
- **Condition, 2 (Swanson, McEliece, 1988):** "any MIU theorem should start with M followed by an arbitrary word in I's and U's"

MU puzzle (cont.)

- **Question:** How to solve it automatically?

MU puzzle (cont.)

- **Question:** How to solve it automatically?
- **Answer:** Let's apply classical FO logic ...

MU puzzle (cont.)

- **Question:** How to solve it automatically?
- **Answer:** Let's apply classical FO logic ...
  - Fully automated solution of the puzzle
  - Puzzle is considered as infinite state safety verification problem
  - Generic Finite Countermodels Method (FCM) is used

Back to MU puzzle: Logic encoding

FO theory MIU:

1. $(x * y) * z = x * (y * z)$ (associativity of concatenation);
2. $e * x = x$;
3. $x * e = x$;
4. $T(M * I)$ (MI is a theorem of MIU);
5. $T(x * I) \rightarrow T(x * I * U)$ (rule I of MIU);
6. $T(M * x) \rightarrow T(M * x * x)$ (rule II of MIU);
7. $T(x * I * I * I * y) \rightarrow T(x * U * y)$ (rule III of MIU)
8. $T(x * U * U * y) \rightarrow T(x * y)$ (rule IV of MIU)

Back to MU puzzle: Logic encoding (cont.)

Proposition

If $w \in L_{MIU}$ *then* $MIU \vdash T(t_w)$

Corollary

- *If* $T(t_S)$ *is not FO provable from* T_{MIU}*, that is* $T_{MIU} \nvdash_{FO} T(t_S)$ *then* $S \notin L_{MIU}$*;*
- *For any non-ground term* $t(\bar{x})$ *in vocabulary* $\{*, M, I, U\}$ *over the set of variables* X*, if* $T_{MIU} \nvdash_{FO} \exists \bar{x} T(t(\bar{x}))$ *then none of* S *such that* t_S *is a ground instance of* $t(\bar{x})$ *belongs to* L_{MIU}*.*

Finite countermodels

Now to show $MIU \nvdash T(M * U)$ we are looking for

- Finite countermodels for $MIU \rightarrow T(M * U)$, or equivalently, for
- Finite models for $MIU \wedge \neg T(M * U)$

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 3 is found in less than 0.01s. The property is proven!

Finite countermodels

Now to show $MIU \nvdash T(M * U)$ we are looking for

- Finite countermodels for $MIU \rightarrow T(M * U)$, or equivalently, for
- Finite models for $MIU \wedge \neg T(M * U)$

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 3 is found in less than 0.01s. The property is proven!

CounterModel as Invariant

The domain D of the model is a three element set $\{0, 1, 2\}$.
Interpretations of constants: $[I] = [M] = 0$, $[U] = 1$.
Interpretation of the predicate T: $[T] = \{1, 2\}$.
The interpretation of the binary function $*$ is given by the following table

	0	1	2
0	2	0	1
1	0	1	2
2	1	2	0

Invariant property which holds for any MIU theorem w:

$$[t_w] \in [T] = \{1, 2\}$$

Notice that $[t_{MU}] = 0 * 1 = 0 \notin [T]$

CounterModel as Invariant (cont.)

In summary

- The interpretation $[*]$ above defines the set of strings $L_{\mathcal{M}} = \{s \mid [t_s]_{\mathcal{M}} \in \{1, 2\}\}$ for which
  - $L_{MIU} \subseteq L_{\mathcal{M}}$
  - $MU \notin L_{\mathcal{M}}$
- Thus, $L_{\mathcal{M}}$ is an invariant separating the theorems of MIU system and the string in question, MU

CounterModel as Invariant (cont.)

In summary

- The interpretation $[*]$ above defines the set of strings $L_{\mathcal{M}} = \{s \mid [t_s]_{\mathcal{M}} \in \{1, 2\}\}$ for which
  - $L_{MIU} \subseteq L_{\mathcal{M}}$
  - $MU \notin L_{\mathcal{M}}$
- Thus, $L_{\mathcal{M}}$ is an invariant separating the theorems of MIU system and the string in question, MU
- It is easy to see also that the invariant is a *regular* language

CounterModel as Invariant (cont.)

In summary

- The interpretation $[*]$ above defines the set of strings $L_{\mathcal{M}} = \{s \mid [t_s]_{\mathcal{M}} \in \{1, 2\}\}$ for which
  - $L_{MIU} \subseteq L_{\mathcal{M}}$
  - $MU \notin L_{\mathcal{M}}$
- Thus, $L_{\mathcal{M}}$ is an invariant separating the theorems of MIU system and the string in question, MU
- It is easy to see also that the invariant is a *regular* language
- Interestingly, $L_{\mathcal{M}} \neq L_{MIU}$ as, for example, $[M * M] = 2 \in [T]$ hence $MM \in L_{\mathcal{M}}$ but $MM \notin L_{MIU}$.

Let us search for countermodels for $MIU \to T(M * M)$.
Mace4 finds a countermodel $\mathcal{M}'$ of size 2, with the domain $\{0, 1\}$, the interpretations of constants M, I and U as $1, 0$ and 0, respectively; the interpretation $[T]$ of $T = \{1\}$. the interpretation of * is given by the table

```
[*]    0   1
          ----
       0 |0,1
       1 |1,0
```

The corresponding invariant $\{s \mid [t_s]_{\mathcal{M}'} = 1\}$ captures the "oddness" of M count in strings, which is sufficient to separate MM from L_{MIU}.

Subsets of configurations in FCM proofs

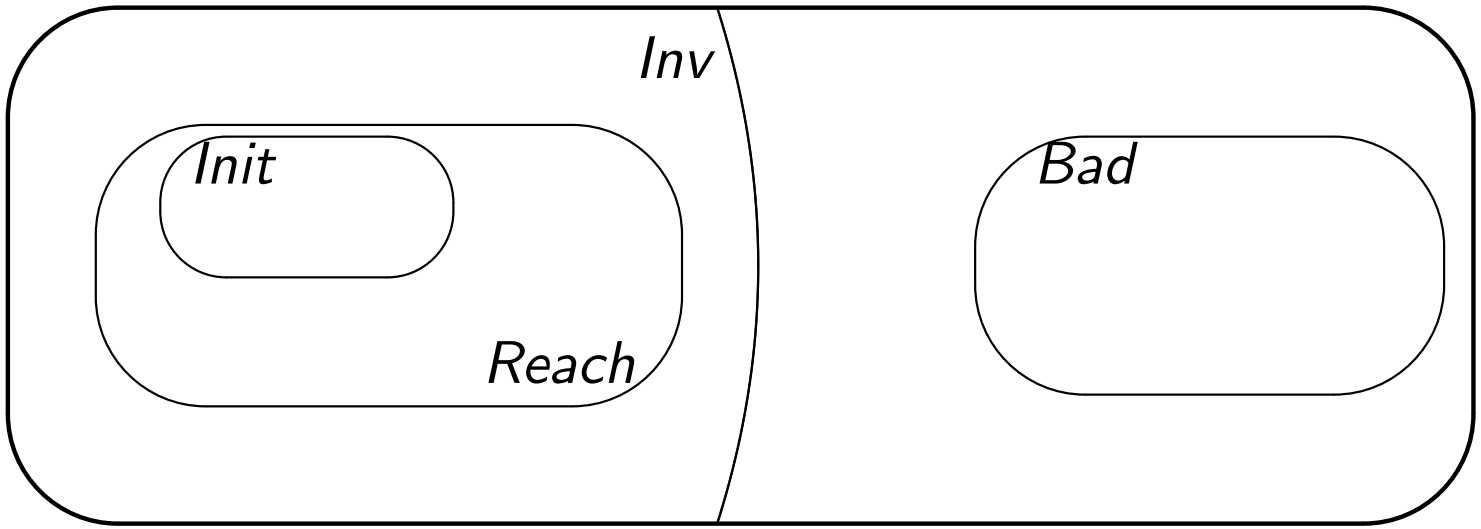


Figure: Subsets of configurations in general position

MU puzzle via formal verificaltion

- MU puzzle was considered as an example in
  E. M. Clarke, A. Fehnker, Z. Han, B. Krogh, J. Ouakine,
  Abtsraction and Counterexample-Guided Refinement in Mpdel Checking of Hybrid System, 2002
- It has been formally verified that MU is not a theorem of MIU, but the proof was not fully automated and required "a good deal of insight'

MU puzzle via formal verificaltion

- MU puzzle was considered as an example in
  E. M. Clarke, A. Fehnker, Z. Han, B. Krogh, J. Ouakine,
  Abtsraction and Counterexample-Guided Refinement in Mpdel Checking of Hybrid System, 2002
- It has been formally verified that MU is not a theorem of MIU, but the proof was not fully automated and required "a good deal of insight'
- Our FCM based verification was fully automated and did not require any insight!

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.

Further developements

AL (2009-...)

- 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
- Generic MACE4 finite model finder by W.McCune has been successfully used to verify above systems

Case Study I: 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

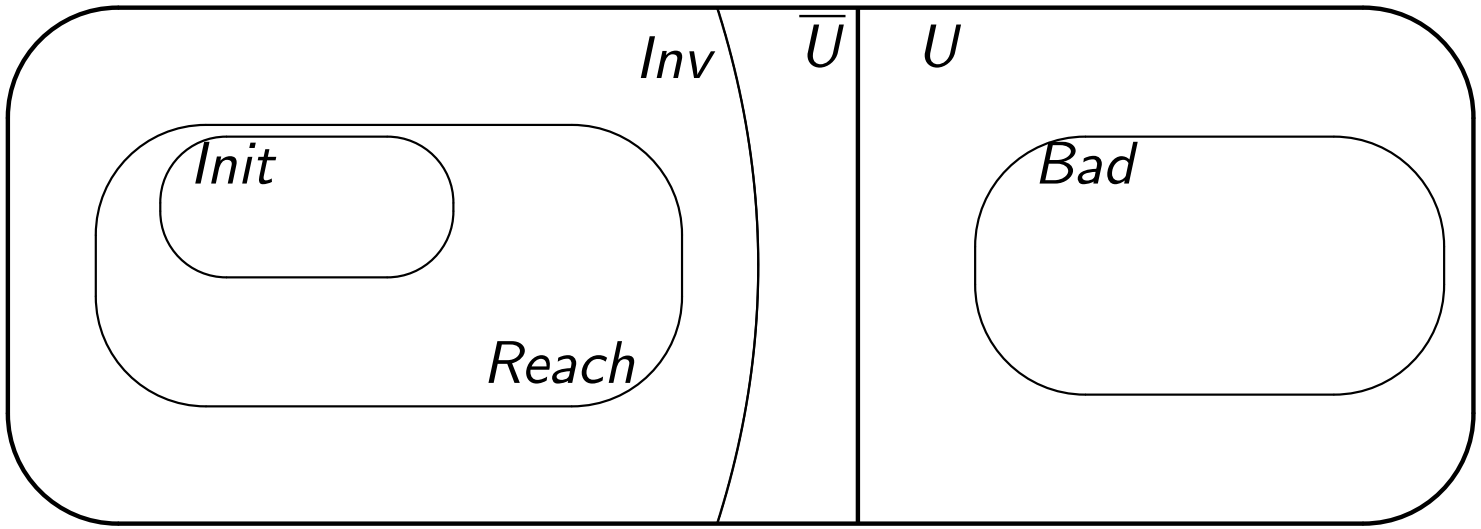


Figure: Subsets of configurations in general position

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.

Case Study II

- Automated verification via finite countermodel finding can be applied to the various parameterized systems

Case Study II

- Automated verification via finite countermodel finding can be applied to the various parameterized systems
- Parameterized cache coherence protocols specified in terms of Extended FSM

Case study II: Caches and cache coherence

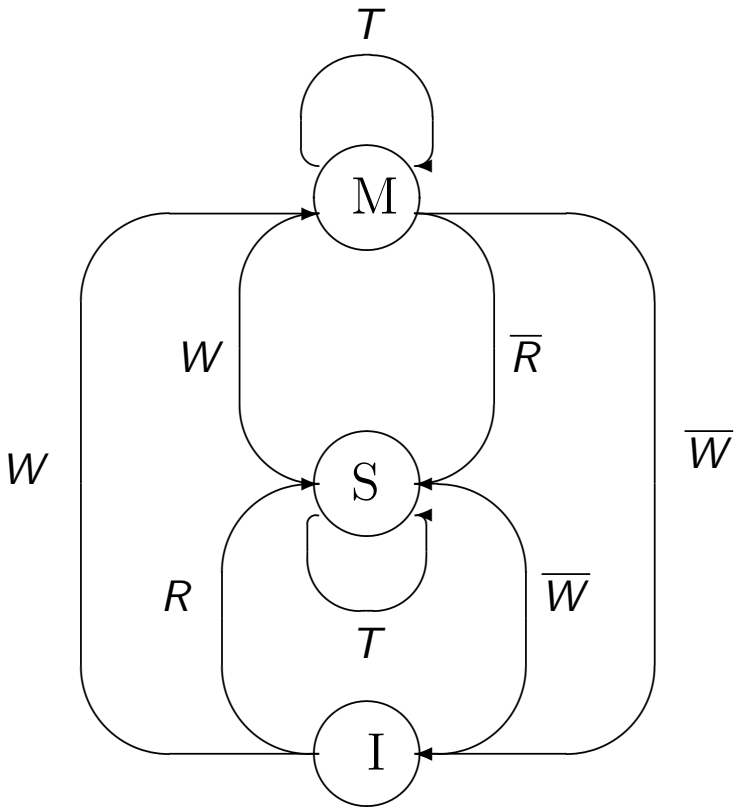
- Cache coherence protocols play an important role in models of shared memory multiprocessor systems.
- Typically, every individual processor has its own private cache memory, which is used to hold local copies of main memory blocks;
- While reducing the access time, this approach poses the problem of cache consistency:
  - one has to ensure that the copies of the *same* memory block in the caches of *different* processors are consistent.

Automata model

Abstracting from low-level implementation details
Cache coherence protocols are

- families of *identical* finite state machines, with the number of the machines being a parameter, together with
- the means of communication:
  - Broadcast: if one automaton makes a transition (an action) a, then it is required that *all* other automata make a complementary transition (reaction) $\bar{a}$
  - Rendevouz: if one automaton makes an action then some other automaton makes a reaction;
- the computation is assumed to be non-deterministic, i.e. at every step one automaton is chosen to make one of the available actions.

MSI Protocol

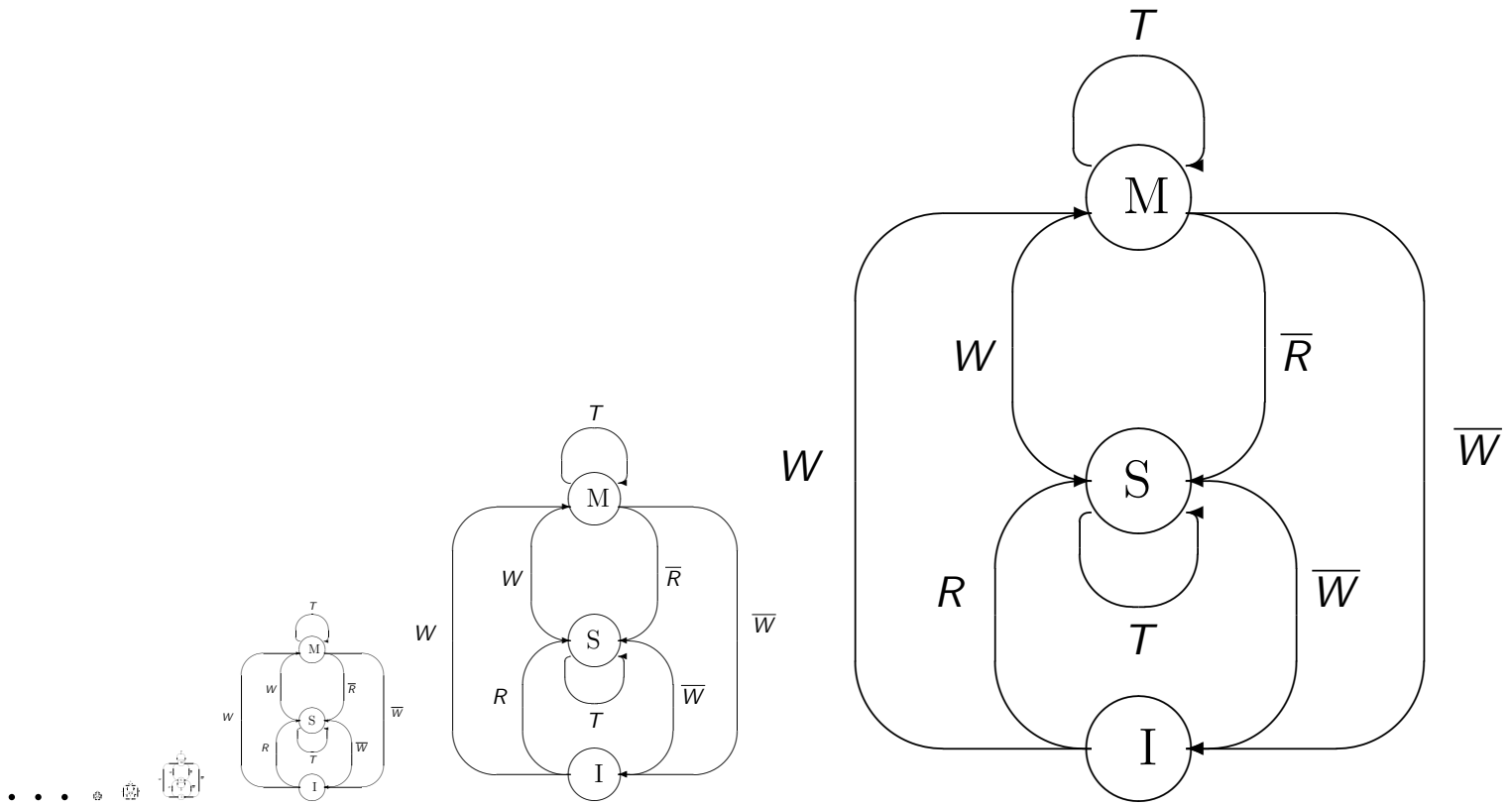


Correctness conditions:

- No two automata are simultaneously in the states S and M.
- No two automata are simultaneously in the state M

$$\tau(I, W) = M \quad \tau(I, R) = S \quad \tau(S, T) = S \quad \tau(M, T) = M$$
$$\tau(I, \overline{W}) = I \quad \tau(I, \overline{R}) = I \quad \tau(I, \overline{T}) = I$$
$$\tau(S, \overline{W}) = I \quad \tau(S, \overline{R}) = S \quad \tau(S, \overline{T}) = S$$
$$\tau(M, \overline{W}) = I \quad \tau(M, \overline{R}) = S \quad \tau(M, \overline{T}) = M$$

MSI global machine



Verification of parameterized protocols

- We would like verify the properties like

  If global machine for MSI of the dimension n starts in a global state with all automata in local states I then during any possible run

  - No two automata are simultaneously in the states S and M.
  - No two automata are simultaneously in the state M

- We would like to verify this property for all n.

Counting abstraction

- In protocols the automata are assumed identical $\Rightarrow$ there is a lot of symmetry in their behaviour;
- Counting abstraction: keep track only of the *numbers of automata* in every possible (local) states.
  - For a broadcast protocol $\mathcal{P} = \langle Q, \Sigma, \bar{\Sigma}, \tau \rangle$, *configuration* of $\mathcal{P}$ is a function $c : Q \to N$;
  - Intuitively, $c(s)$ indicates how many processes are in the local state s
  - If $Q = \{s_1, \ldots, s_n\}$ then with any global state (of any dimension) one may associate configuration, presented as vector $(c(s_1, \ldots, c(s_n)) \in N^n$.

Parameterized configurations

- A parameterized configuration $\equiv$ a set of configurations;
- It can be specified by
  - a vector with indeterminate values (variables): $(1, x, 5, y)$ represents a set of configurations $\{(1, x, 5, y) \mid x, y \in N\}$;
  - a vector with arithmetical terms: $(1, x+1, 5, y+3)$ represents a set of configurations $\{(1, x+1, 5, y+3) \mid x, y \in N\}$;
  - more generally, by a set of arithmetical constraints: $(c(s1) = 3) \wedge (c(s3) > 1)$

Extended FSM

Counting abstraction maps global machines for protocols into a variant of Extended FSM (Cheng, Krishnakumar 1997)

- States of EFSM are non-negative integer vectors;
- Transitions are guarded linear transformations;
- Guards are linear constraints.

Extended FSM

Counting abstraction maps global machines for protocols into a variant of Extended FSM (Cheng, Krishnakumar 1997)

- States of EFSM are non-negative integer vectors;
- Transitions are guarded linear transformations;
- Guards are linear constraints.

Case study: EFSM for MSI

From the paper by E.A. Emerson and V. Kahlon (2003):

(PrWr1) $invalid \geq 1 \rightarrow invalid' = invalid + modified + shared - 1, modified' = 1, shared' = 0.$

(PrWr2) $shared \geq 1 \rightarrow invalid' = invalid + modified + shared - 1, modified' = 1, shared' = 0.$

(PrRd) $invalid \geq 1 \rightarrow invalid' = invalid - 1, modified' = 0, shared' = 1 + shared + modified.$

- The parameterized initial configuration is expressed as: $invalid \geq 1$, $modified = 0$, $shared = 0$
- The potentially unsafe states:
  - $invalid \geq 0$, $modified \geq 1$, $shared \geq 1$
  - $invalid \geq 0$, $modified \geq 2$, $shared \geq 0$

Encoding of EFSM for MSI

Let Φ be a conjunction of the following formulae

- $R(i(x), y, z) \rightarrow R(plus(plus(x, y), z), i(0), 0)$
- $R(x, y, i(z)) \rightarrow R(plus(plus(x, y), z), i(0), 0)$
- $R(i(x), y, z) \rightarrow R(x, 0, i(plus(y, z)))$
- $R(i(x), 0, 0)$
- $plus(0, y) = y$
- $plus(i(x), y) = i(plus(x, y))$

Proposition 1 $\exists n \geq 1((n, 0, 0) \rightarrow^* (k, l, m)) \Rightarrow \Phi \vdash R(i^k, i^l, i^m)$

Verification of MSI

By Proposition 1 it is sufficient to show that

- $\Phi \nvdash (\exists x \exists y \exists z (R(x, i(y), i(z)) \vee R(x, i(i(y)), z))$

Verification of MSI

By Proposition 1 it is sufficient to show that

- $\Phi \nvdash (\exists x \exists y \exists z (R(x, i(y), i(z)) \vee R(x, i(i(y)), z))$
- Now apply finite model finder MACE4

Verification of MSI

By Proposition 1 it is sufficient to show that

- $\Phi \nvdash (\exists x \exists y \exists z (R(x, i(y), i(z)) \vee R(x, i(i(y)), z))$
- Now apply finite model finder MACE4
- A finite model for $\Phi \wedge \neg(\exists x \exists y \exists z (R(x, i(y), i(z)) \vee R(x, i(i(y)), z))$ is found in a less than 0.01 sec of CPU time.

Verification of MSI

By Proposition 1 it is sufficient to show that

- $\Phi \nvdash (\exists x \exists y \exists z (R(x, i(y), i(z)) \vee R(x, i(i(y)), z))$
- Now apply finite model finder MACE4
- A finite model for $\Phi \wedge \neg(\exists x \exists y \exists z (R(x, i(y), i(z)) \vee R(x, i(i(y)), z))$ is found in a less than 0.01 sec of CPU time.
- The protocol is verified.

Experimental results: cache coherence protocols

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

Case study III: Applications to Physical Security

Gold Thief Problem (Kelly, Pearce 2010)
In Gold Thief domain we assume there is thief who may try to steal some gold from a safe:

- The safe is in the room equipped with a light, which may be toggled between "on" and "off"; and with a security camera which may detect thief if the light is "on";
- the safe may be open or closed and the gold can be stolen only if the safe is open;
- It is possible for the thief to crack the safe and force it open, but only if the light is on.

The problem is to show that the gold can not be stolen without thief being detected.

Gold Thief

While the problem was introduced as a case for formalizing reasoning about actions, it has clear security flavor and can be seen as a very simple and rudimentary security scenario.

Gold Thief (cont.)

We take the following axioms for basic action theory for this case from (Kelly, Pearce 2010):

Successor state axioms D_{ssa}:

- $Stolen(do(a, s)) \equiv a = takeGold \vee Stolen(s)$
- $SafeOpen(do(a, s)) \equiv a = crackSafe \vee SafeOpen(s)$
- $LightOn(do(a, s)) \equiv a = (toggleLight \wedge \neg LightOn(s)) \vee (LightOn(s) \wedge a \neq toggleLight)$

Action description axioms D_{ad}:

$$Poss(a, s) \equiv a = toogleLight \vee a = takeGold \wedge SafeOpen(s) \vee a = crackSafe \wedge LightOn(s)$$

$$Undet(a, s) \equiv Poss(a, s) \wedge a \neq toggleLight \wedge \neg LightOn(s)$$

Initial state axiom D_{S_0}:

$$\neg Stolen(s) \& \neg(SafeOpen(s_0)) \vee LightOn(s_0)).$$

Gold Thief (cont.)

In order to apply FCM to prove safety we need to specify/axiomatize the concept of sequence of situations obtained by undetected actions D_{seq}:

- $UndetSeq(s_0)$.
- $(UndetSeq(y) \wedge Undet(x, y)) \rightarrow UndetSeq(do(x, y))$

Claim (adequacy of encoding)
In the "golden thief" domain if safety does not hold, that is thief can steal the gold undetected then
$D_{ssa} \cup D_{ad} \cup D_{seq} \vdash_{FO} \exists z(UndetSeq(z) \wedge Stolen(z))$,
where $\vdash_{FO}$ denotes derivability in first-order logic.

Gold Thief (Cont.)

Now it should be clear that to show safety it is sufficient to demonstrate that

- $D_{ssa} \cup D_{ad} \cup D_{seq} \nvdash_{FO} \exists z(UndetSeq(z) \wedge Stolen(z))$, that is $\nvdash_{FO} D_{ssa} \cup D_{ad} \cup D_{seq} \rightarrow \exists z(UndetSeq(z) \wedge Stolen(z))$.
- We propose to show it by automated search for a finite countermodel, that is a finite model for $D_{ssa} \cup D_{ad} \cup D_{seq} \wedge \nexists z(UndetSeq(z) \wedge Stolen(z))$.
- Now we delegate the last problem to an automated finite model builder program, such as Mace4. Mace4 finds a model in a fraction of the second. The safety is proven.

Further work in Physical Security

- Formalizing physical security procedures in Logic of Moves (Meadows, Pavlovic 2010)

Further work in Physical Security

- Formalizing physical security procedures in Logic of Moves (Meadows, Pavlovic 2010)
- Ongoing research (AL): translate above into the situation calculus and apply FCM

Beyond FCM: limitations of the method

- Can we always apply FCM to establish safety?

Beyond FCM: limitations of the method

- Can we always apply FCM to establish safety?
- No. Here is an example: consider TRS (term rewriting system):
  - $f(x, y) \leftrightarrow f(g(x), g(y))$
  - $f(a, g(x)) \rightarrow a$
  - $f(g(x), a) \rightarrow a$
- Is it true that $f(a, a) \not\rightarrow^* a$?

Beyond FCM: limitations of the method

- Can we always apply FCM to establish safety?
- No. Here is an example: consider TRS (term rewriting system):
  - $f(x, y) \leftrightarrow f(g(x), g(y))$
  - $f(a, g(x)) \rightarrow a$
  - $f(g(x), a) \rightarrow a$
- Is it true that $f(a, a) \not\rightarrow^* a$? Yes! But this can not be established by FCM, for there is no a regular invariant here separating reachable terms and a!

Beyond FCM: limitations of the method

- Can we always apply FCM to establish safety?
- No. Here is an example: consider TRS (term rewriting system):
  - $f(x, y) \leftrightarrow f(g(x), g(y))$
  - $f(a, g(x)) \rightarrow a$
  - $f(g(x), a) \rightarrow a$
- Is it true that $f(a, a) \not\rightarrow^* a$? Yes! But this can not be established by FCM, for there is no a regular invariant here separating reachable terms and a!
- **Challenge:** Extend the method to infinite countermodels!

Conclusion

- We presented FCM method for safety verification of infinite state and parameterized systems

Conclusion

- We presented FCM method for safety verification of infinite state and parameterized systems
- FCM is simple

Conclusion

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

Conclusion

- We presented FCM method for safety verification of infinite state and parameterized systems
- FCM is simple
- FCM is at least as powerful as methods based on monotonic abstraction, RMC, RTMC, tree automata completion techniques in establishing safety
- FCM is efficient in practice (in many cases)

Conclusion

- We presented FCM method for safety verification of infinite state and parameterized systems
- FCM is simple
- FCM is at least as powerful as methods based on monotonic abstraction, RMC, RTMC, tree automata completion techniques in establishing safety
- FCM is efficient in practice (in many cases)