Parameterized Verification of Deadlock Freedom in Symmetric Cache Coherence Protocols

Brad Bingham\textsuperscript{1}  
Jesse Bingham\textsuperscript{2}  
Mark Greenstreet\textsuperscript{1}

\textsuperscript{1}University of British Columbia, Canada  
\textsuperscript{2}Intel Corporation, U.S.A.

November 2, 2011

FMCAD
Outline

1. What is Deadlock-Freedom?
2. Mixed Abstractions for Parameterized Systems
3. Tightening Mixed Abstractions
4. Results
“Is it deadlock-free?” ≡ “Is there a path from each reachable state to a quiescent state?”

- “quiescent” ≡ “nothing is pending”
- In CTL: $\text{AG EF } q$ (more generally, $\text{AG } (p \rightarrow \text{EF } q)$)
- Cheap to model check; rules out some liveness bugs; avoids fairness
A system $S = (S, I, T)$ is a tuple of states $S$, initial states $I$ and transitions $T$.

A parameterized system is a mapping from the naturals to systems. $S(N) = (S(N), I(N), T(N))$.

- In cache coherence protocols, the parameter might correspond to “number of caches”, “number of address”, “length of some buffer”, etc. In our examples, it’s “number of caches”.

Verifying a safety property of $S(N)$ for all $N$ is algorithmically undecidable.

Previous work addresses this problem. One promising approach is based on compositional reasoning (CEGAR + Human Ingenuity).

- [McMillan99], [Chou+04], [O’Leary+09]
Parameterized Cache

Symmetric

cache 1  cache 2  \ldots  cache N

Interconnect

directory
Parameterized Cache Abstraction

- Finite-state, overapproximate abstraction of $S(N)$ for all $N > 2$
- Suitable for model checking
Abstraction Relation

Concrete System $S(N)$
Reachable states: 🟦

Abstract System $A$
Reachable states: 🟤
Abstraction Relation

Concrete System $\mathcal{S}(N)$
- Reachable states: 
- Overapproximation:

Abstract System $\mathcal{A}$
- Reachable states:

Abstraction allows us to infer concrete safety properties 😊
Abstraction Relation

Concrete System $S(N)$
- Reachable states: 😊
- Overapproximation: 💔
- Quiescent states: 😬

Abstract System $A$
- Reachable states: 😌
- Quiescent states: 😬

paths don’t (necessarily) concretize

- ✓ Abstraction allows us to infer concrete safety properties 😊
- ❌ Cannot infer concrete deadlock-freedom properties 😞

Bingham, Bingham, Greenstreet (UBC, Intel)  Parameterized Deadlock Freedom  November 2011  7 / 20
Underapproximate Transitions

Suppose \((s, s')\) is an abstract transition where every reachable state in the concretization of state \(s\) has a path to some state in the concretization of state \(s'\).

This transition is called **underapproximate**.
A Mixed Abstraction [LT88][Dams+97] is like an abstract transition system, but has two sets of transitions: overapproximate ($O$) and underapproximate ($U$).

Model checking $\text{AG}(p \rightarrow \text{EF} q)$ in mixed abstraction $\mathcal{M}$: for each $O$-reachable $p$-state, find a $U$-path to some $q$-state.

**Theorem**

If $\mathcal{M} \models \text{AG}(p \rightarrow \text{EF} q)$, then $\mathcal{S}(N) \models \text{AG}(p \rightarrow \text{EF} q)$. 
What if model checking fails?

1. Perhaps $O$ is too weak
   - State $s$ has no reachable concretization in $S(N)$
   - Remedied by strengthening $O$ (covered by previous literature in parameterized safety)

2. Perhaps $U$ is too strong
   - A $U$-path from $s$ gets “stuck” before a $q$-state is reached
   - Proving that transitions are underapproximate is not addressed by extensive previous work; this is our focus
Assume a symmetric, parameterized system $S(N)$ expressed with guarded commands (or “rules”); assume an overapproximate abstraction of $S(N)$
  
  - Some restrictions to syntactic form

Use the abstraction as a starting point for the mixed abstraction

**Approach:** Use syntactic analysis to find “trivially” underapproximate transitions $U$

**Then:** Prove selected guarded commands of $O$ are in fact underapproximate by leveraging symmetry and model checking the mixed abstraction.
  
  - The approach depends on the syntactic form of the rule
  - All of our methods rely on “path symmetry”
Concrete States

parametric variables, ranging over

\[ P \times G \times L[1] \times L[2] \times L[3] \times \cdots \times L[N] \]

- global boolean variables ranging over \( \{T, F\} \)
- \( L[i] \) symmetric local variables ranging over \( \{T, F\} \)

Bingham, Bingham, Greenstreet (UBC, Intel)  
Parameterized Deadlock Freedom  
November 2011 12 / 20
Abstract States

HIDDEN

parametric variables, ranging over

\{1, 2, \ldots, N\} \quad \{1, 2, Other\}

global boolean variables
ranging over \{T, F\}

L[i] symmetric local variables
ranging over \{T, F\}
(Symmetric) Guarded Commands

Guard \Rightarrow \text{Command}

rule r_1 fires

ptr = 1
\times
G \in A
\times
L[1] \in B
\times
L[2]
\times
L[3]
\times
\vdots
\times
L[N]

rule r_3 fires

ptr = 1
\times
G' \times
L'[1]
\times
L[2]
\times
L[3]
\times
L[N]

ptr = 3
\times
G \in A
\times
L[1]
\times
L[2]
\times
L[3] \in B
\times
L[N]

ptr = 3
\times
G'
\times
L[1]
\times
L[2]
\times
L'[3]
\times
L[N]
(Symmetric) Guarded Commands

rule $r_1$ fires

rule $r_3$ fires

underapprox!

not sure...

HIDDEN BY ABSTRACTION
Abstracted Local State: \( L[\text{ptr}] \in B \land G \in A \)
Abstracted Local State: $L[\text{ptr}] \in B \land G \in A$

indistinguishable in abstraction  rule $r_3$ fires

HIDDEN BY ABSTRACTION
Abstracted Local State: $L[\text{ptr}] \in B \land G \in A$

indistinguishable in abstraction  rule $r_3$ fires

Model Checking Mixed Abstraction

HIDDEN BY ABSTRACTION

implied path

path symmetry
Abstracted Universal Quantifier: \( G \in A \land \forall i. \ L[i] \in B \)
Abstracted Universal Quantifier: $\mathbf{G} \in \mathbf{A} \land \forall i. \mathbf{L}[i] \in \mathbf{B}$

indistinguishable in abstraction

$r_1$ fires

HIDDEN BY ABSTRACTION
Abstracted Universal Quantifier: \( G \in A \land \forall i. L[i] \in B \)

indistinguishable in abstraction

Model Checking Mixed Abstraction

\( r_1 \) fires

implied path

path symmetry

HIDDEN BY ABSTRACTION
German and Flash cache coherence protocols
Proved “For any number of caches, the system can always clear the communication channels and directory is not in a waiting state”
Overapproximate transitions from Murϕ models of strengthened abstractions borrowed from [Chou+04]
Underapproximate transitions proven “on-demand”
- Some transitions are trivially underapproximate by syntactic analysis
- Others are proven underapproximate with our methods, when the model checker indicates a rule will help, i.e., enabled transitions of O at s’

\[
\text{initial state} \quad \begin{array}{c}
0 \\
\rightarrow
\end{array} \quad \begin{array}{c}
0 \\
\rightarrow
\end{array} \quad \begin{array}{c}
\bullet \\
\bullet \\
\bullet \\
\bullet \\
O
\end{array} \quad \begin{array}{c}
O
\rightarrow
\end{array}
\]

\[
\text{p−state} \quad \begin{array}{c}
s \\
\rightarrow
\end{array} \quad \begin{array}{c}
U \\
\bullet
\rightarrow
\end{array} \quad \begin{array}{c}
\bullet \\
\bullet \\
\bullet \\
\bullet
\end{array} \quad \begin{array}{c}
U
\rightarrow
\end{array} \quad \begin{array}{c}
s’
\end{array}
\]

\[
\text{U−path} \quad \text{NOT a q−state; no U transitions out of s’}
\]
Can this process be automated?

- **YES**: Detection of a “useful” rule to prove underapproximate
- **YES**: Application of model checking for the appropriate reasoning (depends on the form of the guard)
- **UNSURE**: What to do if our tricks fail
- **HOWEVER**: When our tricks don’t work, it’s a sign that the rule may NOT be underapproximate.
- **WHAT THEN?**: Perform some manual strengthening similar to previous work!
Future Work

- **Automation:** As mentioned, in a theorem proving environment.
  - Automatically extract from $O$ the weakest $U$ supported by our methods
- **Other Problems:** Parameterize over addresses? (OpenSPARC)
  - Still symmetric, but guards of rules take different syntactic form
- **Other Properties:** Consider request $req$ and response $resp$:
  - Prove “When $req$ is outstanding, there exists a path to $resp$”
  - $AG\ (req\text{-}pend \rightarrow EF\ resp)$
Presented a tractible method for proving parameterized deadlock-freedom

Builds directly on previous work in parameterized safety ([McMillan99,Chou+04])

**Expectation:** Method offers low-hanging deadlock-freedom result following application of these methods, leveraging a tight overapproximation

Thank-you! Questions?