Introduction to Parameterised Verification

Limits For Automatic Verification Of Finite – State Concurrent System

By APT and KOZEN

SHINU NAIR

Under

Prof. Heike Wehrheim

Advisor: Steffen Ziegert
AGENDA

- Parameterised System
- Mutex
- Need for Verification
- Verification Techniques
- Limits of Parameterised Verification
- Techniques of Parameterised Verification
- Summary
AGENDA

- Parameterised System
- Mutex
- Need for Verification
- Verification Techniques
- Limits of Parameterised Verification
- Techniques of Parameterised Verification
- Summary
PARAMETERISED SYSTEM

- Parallel composition of $n$ symmetric processes.
- All processes executing the same program.
- A condition that should satisfy the system.
AGENDA

- Parameterised System
- Mutex
- Need for Verification
- Verification Techniques
- Limits of Parameterised Verification
- Techniques of Parameterised Verification
- Summary
For n >= 2 Processes
Critical Region: No two processes accessing simultaneously
For all i, j ∈ {1, 2, 3, 4 .......}
i ≠ j : ∩ ¬(CRi ⊆ CRj)
PETESENRO's ALGORITHM

- This is a generalisation for n >= 2 processes
- Simple Solution for n = 2 processes
/* trying protocol for P1 */
Q1 := true;
TURN := 1;
wait until not Q2 or TURN := 2;
CRITICAL SECTION;
/* exit protocol for P1 */
Q1 := false

/* trying protocol for P2 */
Q2 := true;
TURN := 2;
wait until not Q1 or TURN := 1;
CRITICAL SECTION;
/* exit protocol for P2 */
Q2 := false
AGENDA

- Parameterised System
- Mutex
- Need for Verification
- Verification Techniques
- Limits of Parameterised Verification
- Techniques of Parameterised Verification
- Summary
NEED FOR VERIFICATION

- Increasing demand of Software.
- Reliability and Safety of the systems.
- A bug in the system can cause heavy loss.
  - Ariane 5: 500 million dollars
    - overflow error in a conversion
      - floating point to integer
  - Intel Pentium: 500 million dollars
    - error in floating point division
EXAMPLE FOR VERIFICATION PROPERTY

Process \( i \) holds the biggest ticket \( y[i] \) by using a universally-quantified assertion

\[ \forall j \neq i : y[j] < y[i] \]
AGENDA

- Parameterised System
- Mutex
- Need for Verification
- Verification Techniques
- Limits of Parameterised Verification
- Techniques of Parameterised Verification
- Summary
Verification Techniques

- **Formal Verification Technique**
  - Correctness proof by using inexpensive simulation
  - Find tough bugs
  - Applied to software and hardware systems
  - Verify infinite state system

- **Formal Verification Technique falls in 2 major categories**
  - Model Checking
  - Deductive Verification
AGENDA

- Parameterised System
- Mutex
- Need for Verification
- Verification Techniques
- Limits of Parameterised Verification
- Techniques of Parameterised Verification
- Summary
Verification

- What is possible?
  - To verify a parameterised system if the parameter‘s value is known

- What is not possible?
  - To verify a parameterised system if the parameter‘s value is not known
Verification Problem

1. \( (P, \Phi) \)
   - \( P(n) \): Finite State Program
   - \( \Phi(n) \): Temporal Logic Formula

2. Will take less time if \( n = 1 \) or \( n = 2 \)
3. Will take infinitely long time if \( n = 1, 2, 3, 4 \ldots \)
RELATION

\[ Q_R(P, \phi) \equiv \begin{cases} 
\forall n \ (P(n) \text{ satisfies } \phi(n)) & \text{if } (P, \phi) \in R, \\
false & \text{otherwise.} 
\end{cases} \]
UNDECIDABILITY

- To show that $Q_R$

  $$\forall n : P(n) \models \Phi$$

  is Undecidable

- We have to reduce the problem in order to prove the undecidibility.
CONSTRUCTION OF VERIFICATION PROBLEM

- \( M \Rightarrow \) Arbitrary Turing Machine
- \( P(n) \Rightarrow \) Finite State Program

\[
\begin{align*}
\text{begin} \\
\quad \text{flag} & \leftarrow \text{false}; \\
\quad \text{for } i & \leftarrow 1 \text{ to } n \text{ do} \\
\quad & \quad \text{simulate one step of } M \\
\quad \text{od;} \\
\quad \text{if } M \text{ has not yet halted then } \text{flag} & \leftarrow \text{true} \\
\text{end}
\end{align*}
\]
HALTING PROBLEM

Reduced Verification Problem gives raise to Halting Problem

\neg \text{HALT}(M) \iff \forall n \ P(n) \text{ satisfies the formula }

\Box(\text{at end} \rightarrow \text{flag})

\iff (P, \Box(\text{at end} \rightarrow \text{flag})) \in Q_R,

where \Box stands for the usual ‘always’ operator of linear temporal logic.

\neg \text{HALT}(M) \iff \forall n \ P(n) \text{ satisfies the formula } \Diamond \text{ flag}

\iff (P, \Diamond \text{flag}) \in Q_R,

where \Diamond stands for the usual ‘eventually’ operator of linear temporal logic.
IDEA

- To construct a Verification Problem that corresponds to Halting Problem
- Verification Problem is as hard as Halting Problem
- Halting problem is Undecidable : Alan Turing 1936
- Hence Relation $Q_R$ is Undecidable
- Hence difficult to verify finite system with parameters having $n$ values
AGENDA

- Parameterised System
- Mutex
- Need for Verification
- Verification Techniques
- Limits of Parameterised Verification
- **Techniques of Parameterised Verification**
- Summary
Techniques for Parameterised Verification

- Counter Abstraction
- Environment Abstraction
- Monotonic Abstraction
- Bounded Parameterised Verification
- Verification by Invisible Invariants
AGENDA

- Parameterised System
- Mutex
- Need for Verification
- Verification Techniques
- Limits of Parameterised Verification
- Techniques of Parameterised Verification
- Summary
SUMMARY

- What are Parameterised Systems?
- What is Mutex Problem?
  - Peterson's Algorithm for Mutex
- Different Verification Techniques
- What is a Verification Problem?
  - Why is Verification Problem for Parameterised System Undecidable?
  - Reduction of Verification Problem
  - Proving that Verification Problem is as hard as Halting Problem
THANK YOU