Verification of Concurrent Programs

Decidability, Complexity, Reductions.

Ahmed Bouajjani

Paris Diderot University, Paris 7

IFIP WG 2.2 meeting
Lisbon, September 2013

Concurrency at different levels

- Application level:
  - Assumes:

    atomicity, isolation, ... (+ sequential specification...)
- Implementation of concurrent data structures, and system services
  - Performances: overlaps between parallel actions, sharing, etc.
  - Ensures:

    (illusion of) atomicity, isolation ...
  - Assumes:

    Memory model (sequential consistency, causal delivery, etc., or weaker ...)
- Infrastructures
  - Performances: Relaxed memory models (reordering, lossyness, etc.)

Issues at different levels

- Applications
  - Correctness: Program (model) satisfies Specification (of some service)

- Concurrent Implementations
  - Ensuring atomicity (+ specification):

    linearizability (shared concurrent data structures),
    serializability (transactions),
    eventual/causal consistency (distributed data structures), etc.

  - Correctness (w.r.t. a specification) over a relaxed memory model.
  - Robustness against a memory model:

    Given a program P and two memory models $M_1 \leq M_2$,
    $[P]_{M_1} = [P]_{M_2}$*?*

Issues at different levels

- Applications
  - Correctness: Program (model) satisfies Specification (of some service)
  - Issues: Complexity (state-space explosion), Undecidability (recursion+synchronization)
- Concurrent Implementations
  - Ensuring atomicity (+ specification):

    linearizability (shared concurrent data structures),
    serializability (transactions),
    eventual/causal consistency (distributed data structures), etc.

  - Correctness (w.r.t. a specification) over a relaxed memory model.
  - Robustness against a memory model:

    Given a program P and two memory models $M_1 \leq M_2$,

    $$[P]_{M_1} = [P]_{M_2}\,?$$

Issues at different levels

- Applications
  - Correctness: Program (model) satisfies Specification (of some service)
  - Issues: Complexity (state-space explosion), Undecidability (recursion+synchronization)
- Concurrent Implementations
  - Ensuring atomicity (+ specification):

    linearizability (shared concurrent data structures),
    serializability (transactions),
    eventual/causal consistency (distributed data structures), etc.

  - Correctness (w.r.t. a specification) over a relaxed memory model.
  - Robustness against a memory model:

    Given a program P and two memory models $M_1 \leq M_2$,
    $[P]_{M_1} = [P]_{M_2}$*?*

  - Issues: Complexity (huge number of action orders), undecidability (some commutations allow to encode TM! – queues).

Questions

- Limits of decidability?
- Complexity?
- Basic (conceptual/technical) tools?
- General and efficient algorithmic approaches?

Reductions to Basic Models

- Pushdown systems ($\equiv$ Recursive state machines)
- Unbounded Petri nets ($\equiv$ Vector Addition Systems)
- (Lossy) FIFO-channel systems

Reductions to Basic Models

- Pushdown systems (≡ Recursive state machines)
  - Model for sequential programs (with recursive procedures).
  - State reachability is polynomial.

- Unbounded Petri nets (≡ Vector Addition Systems)

- (Lossy) FIFO-channel systems

Reductions to Basic Models

- Pushdown systems ($\equiv$ Recursive state machines)
  - Model for sequential programs (with recursive procedures).
  - State reachability is polynomial.
  - Also useful when concurrent behaviors can be "sequentialized".
- Unbounded Petri nets ($\equiv$ Vector Addition Systems)

- (Lossy) FIFO-channel systems

Reductions to Basic Models

- Pushdown systems (≡ Recursive state machines)
  - Model for sequential programs (with recursive procedures).
  - State reachability is polynomial.
  - Also useful when concurrent behaviors can be "sequentialized".
- Unbounded Petri nets (≡ Vector Addition Systems)
  - Model for dynamic concurrent programs with (an arbitrary number of) finite-state (anonymous) threads.
  - State reachability is decidable (EXPSPACE-complete). Research on efficient algorithms.
- (Lossy) FIFO-channel systems

Reductions to Basic Models

- Pushdown systems ($\equiv$ Recursive state machines)
  - Model for sequential programs (with recursive procedures).
  - State reachability is polynomial.
  - Also useful when concurrent behaviors can be "sequentialized".
- Unbounded Petri nets ($\equiv$ Vector Addition Systems)
  - Model for dynamic concurrent programs with (an arbitrary number of) finite-state (anonymous) threads.
  - State reachability is decidable (EXPSPACE-complete). Research on efficient algorithms.
  - Also useful when recursion (stacks) can be "eliminated" using summarization.
- (Lossy) FIFO-channel systems

Reductions to Basic Models

- Pushdown systems ($\equiv$ Recursive state machines)
  - Model for sequential programs (with recursive procedures).
  - State reachability is polynomial.
  - Also useful when concurrent behaviors can be "sequentialized".
- Unbounded Petri nets ($\equiv$ Vector Addition Systems)
  - Model for dynamic concurrent programs with (an arbitrary number of) finite-state (anonymous) threads.
  - State reachability is decidable (EXPSPACE-complete). Research on efficient algorithms.
  - Also useful when recursion (stacks) can be "eliminated" using summarization.
- (Lossy) FIFO-channel systems
  - Model for message-passing programs, and for weak memory models (to encode various kind of buffers, etc.).

Reductions to Basic Models

- Pushdown systems (≡ Recursive state machines)
  - Model for sequential programs (with recursive procedures).
  - State reachability is polynomial.
  - Also useful when concurrent behaviors can be "sequentialized".
- Unbounded Petri nets (≡ Vector Addition Systems)
  - Model for dynamic concurrent programs with (an arbitrary number of) finite-state (anonymous) threads.
  - State reachability is decidable (EXPSPACE-complete). Research on efficient algorithms.
  - Also useful when recursion (stacks) can be "eliminated" using summarization.
- (Lossy) FIFO-channel systems
  - Model for message-passing programs, and for weak memory models (to encode various kind of buffers, etc.).
  - State reachability is decidable for the lossy model (using the theory of WQO). Highly complex (non-primitive recursive), but ...

Reductions to Basic Classes of Programs

- Code-to-code translations to:
  - Sequential programs
  - Concurrent programs over SC
- As general as possible, regardless from the decidability issue:

  Independent from data types, dynamic creation of threads, etc.
- Decidability and complexity are derived for particular cases

  Finite data domains, etc.

Reductions to Basic Classes of Programs

- Code-to-code translations to:
  - Sequential programs
  - Concurrent programs over SC
- As general as possible, regardless from the decidability issue:

  Independent from data types, dynamic creation of threads, etc.
- Decidability and complexity are derived for particular cases

  Finite data domains, etc.
- Questions
  - When is this possible?
  - How?

Concurrent Programs: Sequentialization

- Context-Bounding [Qadeer, Rehof, 05]

Concurrent Programs: Sequentialization

- Context-Bounding [Qadeer, Rehof, 05]
- Sequentialization under Context-bounding
  - Bounded interface $\Rightarrow$ Use an "Assume-Guarantee approach" to analyze sequentially each thread. [Lal, Reps, 08]

Concurrent Programs: Sequentialization

- Context-Bounding [Qadeer, Rehof, 05]
- Sequentialization under Context-bounding
  - Bounded interface ⇒ Use an “Assume-Guarantee approach” to analyze sequentially each thread. [Lal, Reps, 08]
  - Can be generalized to an arbitrary number of statically generated threads: fix-point calculation over the domain of interfaces. [La Torre, Parlato, Madhusudan, 09]

Concurrent Programs: Sequentialization

- Context-Bounding [Qadeer, Rehof, 05]
- Sequentialization under Context-bounding
  - Bounded interface $\Rightarrow$ Use an "Assume-Guarantee approach" to analyze sequentially each thread. [Lal, Reps, 08]
  - Can be generalized to an arbitrary number of statically generated threads: fix-point calculation over the domain of interfaces. [La Torre, Parlato, Madhusudan, 09]
- What about dynamic creation:

Concurrent Programs: Sequentialization

- Context-Bounding [Qadeer, Rehof, 05]
- Sequentialization under Context-bounding
  - Bounded interface $\Rightarrow$ Use an "Assume-Guarantee approach" to analyze sequentially each thread. [Lal, Reps, 08]
  - Can be generalized to an arbitrary number of statically generated threads: fix-point calculation over the domain of interfaces. [La Torre, Parlato, Madhusudan, 09]
- What about dynamic creation:
  - CBA is decidable but at least as hard as State Reachability in Petri nets. [Atig, B., Qadeer, 09].

    $\Rightarrow$ Sequentialization can not be done precisely for CBA.

Concurrent Programs: Sequentialization

- Context-Bounding [Qadeer, Rehof, 05]
- Sequentialization under Context-bounding
  - Bounded interface $\Rightarrow$ Use an "Assume-Guarantee approach" to analyze sequentially each thread. [Lal, Reps, 08]
  - Can be generalized to an arbitrary number of statically generated threads: fix-point calculation over the domain of interfaces. [La Torre, Parlato, Madhusudan, 09]
- What about dynamic creation:
  - CBA is decidable but at least as hard as State Reachability in Petri nets. [Atig, B., Qadeer, 09].
    $\Rightarrow$ Sequentialization can not be done precisely for CBA.
  - Possible under Delay Bounding [Emmi, Qadeer, 11].

Concurrent Programs: Sequentialization

- Context-Bounding [Qadeer, Rehof, 05]
- Sequentialization under Context-bounding
  - Bounded interface ⇒ Use an "Assume-Guarantee approach" to analyze sequentially each thread. [Lal, Reps, 08]
  - Can be generalized to an arbitrary number of statically generated threads: fix-point calculation over the domain of interfaces. [La Torre, Parlato, Madhusudan, 09]
- What about dynamic creation:
  - CBA is decidable but at least as hard as State Reachability in Petri nets. [Atig, B., Qadeer, 09].
    ⇒ Sequentialization can not be done precisely for CBA.
  - Possible under Delay Bounding [Emmi, Qadeer, 11].
  - General schema: tree traversal + bounded interfaces [B., Emmi, Parlato, 11] (Bounded tree-width behaviors)

Serializability and Linearizability

- Known results: Assume a fixed number of threads
  - Serializability: PSPACE-complete [Alur et al. 96, Farzan et al. 08]
  - Linearizability: in EXPSPACE [Alur et al. 96]

Serializability and Linearizability

- Known results: Assume a fixed number of threads
  - Serializability: PSPACE-complete [Alur et al. 96, Farzan et al. 08]
  - Linearizability: in EXPSPACE [Alur et al. 96]
- Arbitrary number of threads?
  - Reductions to State Reachability:

    (Conflict) Serializability and
    Static Linearizability (i.e., when linearization points are fixed, except for read-only methods).

    PSPACE/EXSPACE-complete for fixed/unbounded number of finite-state threads.
  - Linearizability is undecidable in general.

  [B., Enea, Emmi, Hamza, 13]

Weak Memory Models

- State Reachability over a WMM:
  - Decidable for TSO (and PSO ...) [Atig, B., Burckhardt, Musuvathi, 10]
  - True for unbounded store buffers (and arbitrary number of threads).

Weak Memory Models

- State Reachability over a WMM:
  - Decidable for TSO (and PSO ...) [Atig, B., Burckhardt, Musuvathi, 10]
  - True for unbounded store buffers (and arbitrary number of threads).
  - But as hard as State Reachability in Lossy Fifo-Channel Systems (non-primitive recursive)
  - $\Rightarrow$ Reduction to State Reachability in SC is not possible precisely.

Weak Memory Models

- State Reachability over a WMM:
  - Decidable for TSO (and PSO ...) [Atig, B., Burckhardt, Musuvathi, 10]
  - True for unbounded store buffers (and arbitrary number of threads).
  - But as hard as State Reachability in Lossy Fifo-Channel Systems (non-primitive recursive)
  - $\Rightarrow$ Reduction to State Reachability in SC is not possible precisely.
  - (Code-to-code) translation to State Reachability is possible under "Age-bounding" [Atig, B., Parlato, 12]

    Each write action in a buffer must be executed after at most K context switches.

Weak Memory Models

- State Reachability over a WMM:
  - Decidable for TSO (and PSO ...) [Atig, B., Burckhardt, Musuvathi, 10]
  - True for unbounded store buffers (and arbitrary number of threads).
  - But as hard as State Reachability in Lossy Fifo-Channel Systems (non-primitive recursive)
  - $\Rightarrow$ Reduction to State Reachability in SC is not possible precisely.
  - (Code-to-code) translation to State Reachability is possible under "Age-bounding" [Atig, B., Parlato, 12]

    Each write action in a buffer must be executed after at most K context switches.

- Robustness against TSO:
  - State-robustness as hard as State Reachability in TSO.

Weak Memory Models

- State Reachability over a WMM:
  - Decidable for TSO (and PSO ...) [Atig, B., Burckhardt, Musuvathi, 10]
  - True for unbounded store buffers (and arbitrary number of threads).
  - But as hard as State Reachability in Lossy Fifo-Channel Systems (non-primitive recursive)
  - $\Rightarrow$ Reduction to State Reachability in SC is not possible precisely.
  - (Code-to-code) translation to State Reachability is possible under "Age-bounding" [Atig, B., Parlato, 12]

    Each write action in a buffer must be executed after at most K context switches.

- Robustness against TSO:
  - State-robustness as hard as State Reachability in TSO.
  - Trace-robustness is reducible to State Reachability in SC! [B., Derevenetc, Meyer, 13]
  - Code-to-code translation, for an arbitrary number of threads, unbounded buffers, arbitrary data domain.

Conclusion / questions

- A lot remains to be understood concerning decidability frontiers, complexity, and reducibility to problems such as state reachability in basic models.
- In particular: correctness criteria in the distributed case, weak memory models, etc.
- Generic reductions for general classes of programs and general families of correctness criteria.
- Sequentialization (What is pushdown representable ?) is related to the notion of "bounded tree-width" [La Torre, Parlato, Madhusudan, 11].
- We need a general framework for reasoning about order constraints and their violations. (What is Petri net representable ?)