Parameterized Verification of Distributed Systems

Giorgio Delzanno

DIBRIS, Università di Genova, Italy

AILA, Gargnano, Agosto 2017
Background
Petri Nets: Rendez-vous Synchronization

Rendez-vous

new marking

?a

current marking

!a

new marking

G. Delzanno
Let $M_1 \leq M_2$ if, for any color $c$, the number of $c$-tokens in $M_1$ is less than that in $M_2$

- Input: Markings $M_0$ and $M_1$
- Problem: Is there $M_2$ s.t. $M_1 \leq M_2$ and there is a computation from $M_0$ to $M_2$?
Reachability and Coverability are decidable for Petri Nets
  - Karp-Miller construction solves Coverability
Coverability is decidable for Broadcast Protocols
  - Backward Reachability using upward closed sets of markings is guaranteed to terminate
  - Complexity is non-primitive recursive
Assumption:
algorithm to compute finite representations of upward closed set of predecessors
What if nodes are distributed on a graph?
Broadcast Communication on a Graph

!!a

??a

G. Delzanno
Verification of Distributed Systems
Broadcast Communication on a Graph

!!!a

??a

G. Delzanno
Verification of Distributed Systems
Broadcast Communication on a Graph

!!a
??a

G. Delzanno
Broadcast Communication on a Graph

\[ \text{!!a} \]

\[ \text{??a} \]
An Ad Hoc Network (AHN)\(^1\) is a network of communicating automata distributed over an undirected graph.

- Each node runs a predefined common protocol.
- Synchronous selective broadcast messages.

---

\(^1\)Delzanno, Sangnier, Zavattaro: Parameterized verification of ad hoc networks. CONCUR’10.
For:
- a (possibly infinite) class of graphs $G$ (initial network configurations)

Given:
- an automaton/protocol $P$
- a specification $\varphi$ of target states (e.g. is there a red node?)

Output:
- True iff there is an initial configuration in $G$ s.t., by executing $P$ on each node, the network may reach a configuration satisfying $\varphi$
Expressive Power?

- **Fixed initial configuration:**
  - Finite-state systems that are computationally expensive to verify with existing model checkers like Spin, SMV, Uppaal, Groove.

- **Unknown initial configuration, specification: existence of a ”red” node**
  - Verification becomes undecidable: discovery protocols can be used to build unbounded data structures, e.g., a list of arbitrary length.
Discovery

\[ \text{!!a} \]

\[ \text{??a} \]

\[ \text{a} \]
Discovery

!!a

??a

G. Delzanno
COVER becomes decidable by restricting configurations in specific classes of graphs.

- Fully connected topologies
- $K$-bounded path graphs, i.e., graphs in which all simple paths have length $\leq K$ (fully connected graphs $\neq$ not bounded path graphs)
- Graphs in which when collapsing maximal cliques we still have $K$-bounded paths only
• Transitions are monotone w.r.t. the induced subgraph ordering
• Induced subgraph ordering over $K$-bounded path graph is a well quasi ordering [Ding ’90]
• We can effectively compute predecessors of upward closed sets of graphs (for graphs of size $n$ we need to consider at most graphs of size $n + 1$)
Remark 1: Induced vs Subgraph

A subgraph is a graph formed from a subset of the vertices and edges of a larger graph. A subgraph may include any number of vertices and edges, but does not necessarily include all of the vertices and edges.

An induced subgraph is a subgraph that includes all the edges between the vertices in the subset. In other words, an induced subgraph preserves the connectivity of the original graph within the subset of vertices.

The diagram illustrates this concept with two sets of nodes. The left set represents a subgraph, while the right set represents a subgraph that is also an induced subgraph.
Remark 2: Subgraphs is not OK

subgraph of

!!a
??a

subgraph of

!!a
??a
??a
Remark 3: Induced is not a wqo for arbitrary graphs

infinite sequence of incomparable graphs
Possible Extensions

- no fixed infrastructure, dynamically changing: reconfigurations
- asynchronous communication
- value passing
- failures of nodes and/or communication channels
- time constraints
- permission and access control
- ...

G. Delzanno  Verification of Distributed Systems  19/46
Reconfiguration = the topology of the network can change non-deterministically during an execution (node mobility, link failures/intermittence) \(^2\)

As specification language, we consider cardinality constraints (CC) on the number of processes in a given control state:

\[ \varphi ::= a \leq \#q < b \mid \varphi \land \varphi \mid \varphi \lor \varphi \mid \neg \varphi \]

where \( a \in \mathbb{N} \), \( q \) is a local control state, and \( b \in (\mathbb{N} \setminus \{0\}) \cup \{+\infty\} \).

COVER with no restrictions on the initial number of processes.
- COVER is \( \text{PTime} \)-complete for CC without negation and with only \( \#q \geq 1 \) atoms
  - Keypoint: if two processes can make a transition, any number of processes can do the same transition
  - No need to count: Saturation procedures that collects the set of labels that can be produced by applying transitions
- For CC with \( \#q \geq 1 \) atoms and negation COVER is \( \text{NP} \)-complete.
- COVER is \( \text{PSpace} \)-complete for unrestricted CC.
A variant of AHNs with asynchronous broadcast communication.\(^3\)

Unread messages are kept in local mailboxes.

We consider different disciplines for handling mailboxes:
- **multisets**, to model the loss of the order of incoming messages;
- **lossy FIFO queues**, to model the loss of messages;
- **FIFO queues**, to model perfect communication.

\(^3\)Delzanno, Traverso: A Formal Model of Asynchronous Broadcast Communication. ICTCS’12.
Delzanno, Traverso: Decidability and Complexity Results for Verification of Asynchronous Broadcast Networks. LATA’13.
AHN + Asynch.: Example #1

(with multisets as mailboxes)
AHN + Asynch.: Example #1

(with multisets as mailboxes)
AHN + Asynch.: Example #1

(with multisets as mailboxes)
AHN + Asynch.: Example #1

(with multisets as mailboxes)
AHN + Asynch.: Example #1

(with multisets as mailboxes)
AHN + Asynch.: Example #2

(with FIFO queues as mailboxes)
AHN + Asynch.: Example #3

(with lossy FIFO queues as mailboxes)
To decidability boundaries for COVER we must consider

- the **policy** to handle local mailboxes;
- the **shape** of connection graph;
- the capability to **recognize empty mailboxes** ($\epsilon$-transitions).
COVER for multisets and emptiness test is undecidable

Reduction of halting problem for two-counter machines.

*Proof idea:* The emptiness test can be exploited both for zero-testing and interference detection.
Reduction of halting problem for two-counter machines in which the emptiness test can be exploited both for zero-testing and interference detection

- The encoded protocol is split in two phases: **election** and **simulation**.
- During the election processes choose their role.
- The simulation requires a leader process directly connected to two slaves (one per counter).
Election

- Each node chooses a role and searches for appropriate neighbours accordingly.
- Election only tests for the presence of the required links between nodes with the various roles.
- Messages exchanged during the election can never be consumed afterwards.
- A successful election ends leaving empty mailboxes in the involved nodes.
AHN + Asynch.: Undecidability with Emptiness Test

\begin{align*}
q_0 & \xrightarrow{b(s_1)} r(s_1) \xrightarrow{r(s_2)} \mathcal{P}(\ell_0) \\
q_0 & \xrightarrow{b(c)} \xrightarrow{b(s_2)} r(c) \xrightarrow{q_{S_1}} \\
q_{halt} & \xrightarrow{\tau} r(c) \xrightarrow{r(c)} q_{S_2}
\end{align*}
Simulation

- Counters are encoded (in unary) through messages in the mailboxes.
  - For increment it is sufficient to send a broadcast with a unit.
  - A decrement forces the removal of a unit from the mailbox of the slave.
  - Tests for zero are performed by exploiting tests for emptiness.
Warning

What if other neighbours wake up and start a simulation leading to interferences with the current one?
Warning

What if other neighbours wake up and start a simulation leading to interferences with the current one?

Finalization

- Reminder: we cannot consume messages from the election during the simulation.
- After reaching the target control state, we reset both counters to zero in order to try to empty all mailboxes.
- If all mailboxes are empty the simulation ends successfully, otherwise it blocks just before completing.
\[ x_1 - - \quad x_1 == 0 \quad \ell_f \]

\[ x_2 - - \quad x_2 == 0 \quad \overline{\ell_f} \]

\[ P(\overline{\ell_f}) \quad \epsilon \quad q_{target} \]
## AHN + Asynch.: Results

<table>
<thead>
<tr>
<th></th>
<th>( \text{COVER}^K(M) ) without ( \epsilon )</th>
<th>( \text{COVER}^K(M) ) with ( \epsilon )</th>
<th>( \text{COVER}(M) ) without ( \epsilon )</th>
<th>( \text{COVER}(M) ) with ( \epsilon )</th>
</tr>
</thead>
<tbody>
<tr>
<td>LFIFO</td>
<td>( \text{PTime} )</td>
<td>( \text{PTime} )</td>
<td>( \text{PTime} )</td>
<td>( \text{PTime} )</td>
</tr>
<tr>
<td>Multiset</td>
<td>( \text{PTime} )</td>
<td>undec.</td>
<td>( \text{PTime} )</td>
<td>undec.</td>
</tr>
<tr>
<td>FIFO</td>
<td>undec.</td>
<td>undec.</td>
<td>undec.</td>
<td>undec.</td>
</tr>
</tbody>
</table>

|                  | AHN without \( \epsilon \) / with \( \epsilon \) |
|------------------|--------------------------------|--------------------------------|--------------------------------|--------------------------------|
|                  | AHN                           | LFIFO                         | Multiset                      | FIFO                          |
| Fully connected graphs | ✓                           | \( \text{PTime} \)            | \( \text{PTime}/\text{undec.} \) | undec.                        |
| Arbitrary graphs   | undec.                        | \( \text{PTime} \)            | \( \text{PTime}/\text{undec.} \) | undec.                        |
Broadcast Networks of Register Automata\(^4\).

Each process has local registers:
- with values ranging over \( \mathbb{N} \);
- initialized with fresh values w.r.t. the whole network;
- read-only ones act as process identifiers.

Messages have some payload fields to exchange data.

Upon reception of a message with a payload, processes may:
- test (in)equality w.r.t. local registers;
- store values in local registers;
- ignore (a part of) the payload.

Dynamic network reconfigurations (w.r.t. edges).

---

sender nodes want to keep a route towards some dest node.

Two registers: id and next.

Initial states: sender, idle, dest.
We consider queries expressing graph inclusion patterns:

\[ \varphi ::= q(z) \mid M_i(z) = M_j(z') \mid M_i(z) \neq M_j(z') \mid \varphi \land \varphi \]

where \( z, z' \) are from a denumerable set of variables, \( q \) is a local control state, and \( i, j \in \mathbb{N} \) are register indexes.

- No restrictions on the initial number of processes or network topology.
$\text{ready}(z_1) \land \text{ready}(z_2) \land M_{id}(z_1) = M_{next}(z_2) \land M_{next}(z_1) = M_{id}(z_2)$
Explore decidability boundaries for COVER:

- restrictions on the **number of registers** in each node;
- restrictions on the **number of fields** in each message;
- **static topology** vs **dynamic reconfigurations**.
Without network reconfigurations (fully connected case):
- 1 RW registers, 1 payload field $\implies$ non-elementary
- 1 RO + 1 RW registers, 1 payload field $\implies$ undecidable

With reconfigurations:
- $k \geq 0$ RW registers, 1 payload field $\implies$ PSPACE-complete
- 1 RO + 1 RW registers, 2 payload fields $\implies$ undecidable
For each restricted model for which it’s possible, we provide a list-builder protocol.

A list-builder uses registers as pointers to build linked lists of nodes.

It is enough to define a list-builder to prove undecidability:
  - the protocol is extended in order to encode a 2-counter machine;
  - only the subset of features common to all models is used.
network reconfigurations, 2 registers, 2 fields:

\[ b(s, id, id) \rightarrow r(a, \downarrow x, ?id) \rightarrow b(sa, id, x) \rightarrow q_t \]

\[ q_0 \rightarrow r(s, \downarrow x, \ast) \rightarrow b(a, id, x) \rightarrow r(sa, ?x, ?id) \rightarrow q_i \rightarrow \tau \rightarrow q_h \]

\[ q_z \rightarrow b(sa, id, x) \rightarrow r(a, \downarrow x, ?id) \rightarrow b(s, id, id) \]
network reconfigurations, 2 registers, 2 fields:

\[ b(s, id, id) \rightarrow r(a, \downarrow x, ?id) \rightarrow b(sa, id, x) \rightarrow q_t \]

\[ q_0 \rightarrow r(s, \downarrow x, *) \rightarrow b(a, id, x) \rightarrow r(sa, ?x, ?id) \rightarrow q_i \rightarrow \tau \rightarrow q_h \]

\[ q_z \rightarrow b(sa, id, x) \rightarrow r(a, \downarrow x, ?id) \rightarrow b(s, id, id) \rightarrow \implies \text{undecidability} \]
More Recent Work

Concurrent model for an abstraction of the life cycle of Android Components

- Activities are abstracted into finite state automata with local registers (to model data)
- Callbacks are defined via value passing messages
- Permissions are statically defined via a dependency graph between definitions

We study decidability boundaries for the following problems

- Violation of the permission model during component execution (run time errors)
- Detection of conflicts due to data exchanged by components with incompatible permissions

Decidability using History WSTS in which monadic predicates maintain footprints of data along a computation
...we studied the impact on decidability and complexity for several basic features of ad hoc protocols, considering also the interplay between them:

- synchronous communication,
- asynchronous communication,
- local clocks,
- dynamic network reconfigurations,
- data.

...decidability is hard to achieve in general, but not impossible:

- search for good compromises between features of the model, approximations, and expressivity;
- there is a relation between asynchronous communication and reconfigurations, but the latter is easier to handle.