Verification of Communication Protocols with Messages Carrying Values

Tristan Le Gall

joint work with
Bertrand Jeannet and Thierry Jéron

Vertecs team

IRISA/INRIA Rennes
Communication protocols are widely used with the development of the internet and other networks.

Formal verification uses models like Messages Sequence Charts (MSCs), Communicating Finite-State Machines (CFSMs).

Our goal: the verification of protocols modeled by an extension of CFSM, using the abstract interpretation framework.

This work can be applied to process/components of a system using queues or large buffers, Kahn networks, etc.
The CFSM model

(a) Client

(b) Queues

(c) Server

(d) Global CFSM: product of client and server processes
The CFSM model

(a) Client

1!open
1!close
2?disconnect

(b) Queues

close  open
disconnect

(c) Server

1?open
1?close
2!disconnect

(d) Global CFSM: product of client and server processes

\[ C = \{(0, 0), (1, 0), (0, 1), (1, 1)\} \]
The CFSM model

(a) Client

(b) Queues

(c) Server

(d) Global CFSM: product of client and server processes

$\Sigma = \{\text{open, close}\} \cup \{\text{disconnect}\}$
The CFSM model

(a) Client

(b) Queues

(c) Server

(d) Global CFSM: product of client and server processes

initial location $c_0 = (0, 0)$
The CFSM model

(a) Client

(b) Queues

(c) Server

(d) Global CFSM: product of client and server processes

An input: 1?o
The CFSM model

(a) Client

(b) Queues

(c) Server

(d) Global CFSM: product of client and server processes

An output: 1!o
The CFSM model

(a) Client

(b) Queues

(c) Server

(d) Global CFSM: product of client and server processes

A state of the CFSM: a location + contents of all queues
Problematics

- Verification of safety properties
- Main issue: reachability analysis
- Undecidable in the general case
- Our solution: compute an over-approximation of the reachability set
Outline

- Introduction: model and problematics
- Verification of CFSM
- Limitations and new model
- Representation of queues with messages carrying values
- Application to the verification of symbolic CFSM
- Conclusion
CFSM with a single queue

- \( \Sigma \): alphabet of messages
- a content of a queue: a word \( w \in \Sigma^* \)
- \( C \rightarrow L(\Sigma^*) \)
- Operational semantics in terms of operations on languages:
  \[
  (c_1, L) \xrightarrow{\langle c_1, !a, c_2 \rangle} (c_2, L.a)
  \]
- Reachability analysis: a fixpoint equation

- Fixpoint equation \( L = \varepsilon \cup L.a \cup L/a \)
- Solution: \( L = a^* \)
Main idea: work with regular over-approximations of the content of the queue

See the regular languages as an abstract lattice \((\text{Reg}(\Sigma), \subseteq)\)

Compute an over-approximation of the least fix-point with iterations

\[
L_0 = \varepsilon \\
L_{i+1} = L_i \cup L_i a \cup L_i / a
\]

Use a widening operator so that the computation terminates
Widening operator for regular languages (1)

- Working on the Minimal Deterministic Automaton (MDA) $M_L$
- Quotient automaton $\tilde{M}_L = M_L / \approx_k$ (fusion of states)

$\approx_k :$ auto-bisimilarity of depth $k$

$\rho_k(L) :$ language recognized by this quotient automaton
Widening operator for regular languages (2)

- Widening operator $L_1 \nabla_k L_2 \triangleq \rho_k(L_1 \cup L_2)$
- The following computation terminates and gives an over-approximation of the reachability set:
  \[
  L_{i+1} = L_i \nabla_k (L_i.a \cup L_i/a)
  \]
- Result: $L_\infty = a^*$
The client can open and close a session, or be forced to close the session if a disconnect message is received.

The server can ask for a client to terminate his session.
Analysis of the connexion/deconnexion Protocol

Analysis with dependance

<table>
<thead>
<tr>
<th>Client/Server</th>
<th>Queue 1 ≠ Queue 2</th>
</tr>
</thead>
<tbody>
<tr>
<td>0/0</td>
<td>(co)∗(oc)∗#ε + c(oc)∗#d</td>
</tr>
<tr>
<td>1/0</td>
<td>(co)∗(oc)∗o#ε + (co)∗#d</td>
</tr>
<tr>
<td>0/1</td>
<td>c(oc)∗#ε</td>
</tr>
<tr>
<td>1/1</td>
<td>(co)∗#ε</td>
</tr>
</tbody>
</table>

Analysis without dependance

<table>
<thead>
<tr>
<th>Client/Server</th>
<th>Queue 1</th>
<th>Queue 2</th>
</tr>
</thead>
<tbody>
<tr>
<td>0/0</td>
<td>o∗ + (o∗c)⁺(ε + o⁺ + o⁺c)</td>
<td>d⁺</td>
</tr>
<tr>
<td>1/0</td>
<td>(o∗c)⁺o⁺</td>
<td>d⁺</td>
</tr>
<tr>
<td>0/1</td>
<td>o∗ + (o∗c)⁺(ε + o⁺ + o⁺c)</td>
<td>d⁺</td>
</tr>
<tr>
<td>1/1</td>
<td>o⁺ + o∗(co⁺)⁺</td>
<td>d⁺</td>
</tr>
</tbody>
</table>

- Analysing the queues alltogether gives the exact result
- Analysing each queue independently gives a very bad approximation
Protocol with non-regular reachability set

There is a non-regular reachability set.

- Exact result: \( L_{(0/0)} = a^n \epsilon c^n \epsilon \)
- Relational analysis result:
  \[
  L_{(0/0)} = \epsilon \# \epsilon \# \epsilon + a \# c \# c \# + a a a \# c c c \# \epsilon
  \]
Outline

- Introduction: model and problematics
- Verification of CFSM
- Limitations and new model
- Representation of queues with messages carrying values
- Application to the verification of symbolic CFSM
- Conclusion
The analysis terminates and return an over-approximation of the reachability set
The approximations of the queue contents are quite precise
We defined an abstract lattice for regular languages
But the model is not expressive enough
Can we check the (in)equations:

- $A \leq R$
- $S \geq H$
- $MR - R = MS - A$
What shall we do?

- Do not care about real protocols
- Add variables and parameters to the CFSM model and use a similar method on the new model.
New Model : Symbolic CFSM

Each transition has

- a guard : predicate on the value of the variables and the parameter
- a communication action with a parameter \( p \) : emission \(!a(p)\) or reception \(?a(p)\)
- an affectation : gives the new value of the variables
Example of analysis

\[ p = x \]
\[ !a(p) \]
\[ x := x + 1 \]

- Toy example: the producer
- Non-relational analysis: does not keep relation between the value of \( x \) and the values of the messages
- Relational analysis: keep relation between the value of \( x \) and the values of the messages
## Non-relational analysis

<table>
<thead>
<tr>
<th>Computation step</th>
<th>Set of reachable states</th>
</tr>
</thead>
<tbody>
<tr>
<td>init</td>
<td>$[0, 0] \times \varepsilon$</td>
</tr>
<tr>
<td>step 1</td>
<td>$[0, 1] \times a([0, 0])$</td>
</tr>
<tr>
<td>step 2</td>
<td>$[0, 2] \times a([0, 0]) + a([0, 0]) \cdot a([0, 1])$</td>
</tr>
<tr>
<td>step 3</td>
<td>$[0, 3] \times a([0, 0]) + a([0, 0]) \cdot a([0, 1]) + a([0, 0]) \cdot a([0, 1]) \cdot a([0, 2])$</td>
</tr>
<tr>
<td>step 4*</td>
<td>$[0, +\infty[ \times a([0, 1]) + a([0, 1]) \cdot (a([0, +\infty[]))^*$</td>
</tr>
</tbody>
</table>

We lose the relation $p \leq x$. 
### Relational analysis

<table>
<thead>
<tr>
<th>Computation step</th>
<th>Set of reachable states</th>
</tr>
</thead>
<tbody>
<tr>
<td>init</td>
<td>${0 \leq x \leq 0} \land \varepsilon$</td>
</tr>
<tr>
<td>step 1</td>
<td>${0 \leq x \leq 1} \land a({0 \leq p = x - 1})$</td>
</tr>
<tr>
<td>step 2</td>
<td>${0 \leq x \leq 2} \land a({0 \leq p = x - 1}) + a({0 \leq p = x - 2}).a({0 \leq p = x - 1})$</td>
</tr>
<tr>
<td>step 3</td>
<td>${0 \leq x \leq 3} \land a({0 \leq p = x - 1}) + a({0 \leq p = x - 2}).a({0 \leq p = x - 1}) + a({0 \leq p = x - 2}).a({0 \leq p = x - 3}).a({0 \leq p = x - 2}).a({0 \leq p = x - 1})$</td>
</tr>
</tbody>
</table>
| step 4*          | $\{0 \leq x\} a(\{0 \leq p \leq x - 1\}) + a(\{0 \leq p \leq x - 1\}).a(\{0 \leq p \leq x - 1\}) + a(\{0 \leq p \leq x - 1\})$.a(\{0 \leq p \leq x - 1\}))^*
New abstractions

In both cases we need:

- Abstractions for the values of the variables: intervals, polyhedra,
- Representation of languages on the infinite alphabet $\Sigma \times \Omega$
- Automata “with a lattice feature”
Outline

- Introduction: model and problematics
- Verification of CFSM
- Limitations and new model
- Representation of queues with messages carrying values
- Application to the verification of symbolic CFSM
- Conclusion
Lattice structure

- $(\Omega, \sqsubseteq_\Omega)$ an abstract lattice (abstract values of the parameter)
- $\Sigma = \{a_1, \ldots, a_n\}$ finite alphabet of messages
- $\Lambda = \Sigma \times \Omega$ the lattice with
  $$(a_1, P_1) \sqsubseteq (a_2, P_2) \iff a_1 = a_2 \land P_1 \sqsubseteq_\Omega P_2$$
- Example: lattice: $\Lambda = \Sigma \times \mathcal{I}$; value: $a([0, +\infty[)$
Lattice automaton

- Finite automaton with transitions labeled by $\lambda \in \Sigma \times \Omega$

\[ \begin{align*}
  a([0,4]) & \quad a([0,0]) \\
  c([0,0]) & \quad b([0,0]) \\
  b([1,3]) & \quad a([2,3]) \\
  a([4,8]) & \quad c([4,8]) \\
\end{align*} \]

- $q_1 \xrightarrow{\lambda} q_2$ if
  1. $\lambda \neq \perp$
  2. there is a transition $(q_1, \lambda', q_2)$ in the automaton with $\lambda \sqsubseteq \lambda'$

- Accepted words: as for classical finite automata.
- $\text{Reg}(\Lambda)$: languages recognized by a lattice automaton
Algorithms for lattice automata

- **Union**: as for classical finite automata
- **Intersection**:
  \[ a([0,3]) \cap a([2,5]) = a([2,3]) \]
- **Inclusion**: simulation taking into account the lattice structure
- **Determinisation, minimisation, quotient of \( \mathcal{A} \)**: use \( \text{Shape}(\mathcal{A}) \)
Algorithms for lattice automata

- Union: as for classical finite automata
- Intersection:
  \[
  a([0,3]) \cap a([2,5]) = a([2,3])
  \]
- Inclusion: simulation taking into account the lattice structure
- Determinisation, minimisation, quotient of \(\mathcal{A}\): use \(\text{Shape}(\mathcal{A})\)

![Diagram of lattice automata]
Determinisation of lattice automata

Idea of the algorithm:

\[ \text{Shape}(A_0) \text{ is deterministic (so is } A) \]

- \( A' \) is an over-approximation of \( A \):
  \[ L_A \subseteq L_{A'} \]

- \( A' \) is the best approximation:
  \( \forall B \text{ such as } \text{Shape}(B) \text{ is deterministic and } L_A \subseteq L_B, \text{ then } L_{A'} \subseteq L_B \)
Quotient and minimisation

Same principle as before

$\mathcal{L}$ recognized by $\mathcal{A}$. There exists an unique $\mathcal{A}'$:

- $Shape(\mathcal{A}')$ is deterministic and minimal (so is $\mathcal{A}'$)
- $\mathcal{A}'$ is an over-approximation of $\mathcal{A}$:
  $L_\mathcal{A} \subseteq L_{\mathcal{A}'}$
- $\mathcal{A}'$ is the best approximation:
  $\forall \mathcal{B}$ such as $Shape(\mathcal{B})$ is deterministic and minimal, and $L_\mathcal{A} \subseteq L_{\mathcal{B}}$, then $L_{\mathcal{A}'} \subseteq L_{\mathcal{B}}$
If $L_{\mathcal{A}_1} \subseteq L_{\mathcal{A}_2}$:

1. consider the quotient automaton $\mathcal{A}_2/ \simeq_k$

2. If $\mathcal{A}_1$ and $\mathcal{A}_2/ \simeq_k$ have the same shape:

\begin{align*}
\text{a([2,4])} & \quad \text{a([2,7])} & \quad \text{a([2, +\infty])} \\
\text{b([2,4])} & \quad \text{b([0,4])} & \quad \text{b([\infty, 4])}
\end{align*}
Outline

- Introduction: model and problematics
- Verification of CFSM
- Limitations and new model
- Representation of queues with messages carrying values
- Application to the verification of symbolic CFSM
- Conclusion
Non-relational analysis: the lattice of intervals

- The value of each variable is represented by an interval
- Lattice automata on $\Lambda = \Sigma \times \mathcal{I}$
- Abstract semantics:

\[
\begin{align*}
p &= x \\
!a(p) &
\end{align*}
\]

\[
\begin{align*}
x &:= x + 1 \\
x \quad \text{queue} \quad &
\end{align*}
\]

\[
\begin{align*}
[0, 2] &\quad a([0, 1]) &\rightarrow &\quad [1, 3] &\quad a([0, 1]) \cdot a([0, 2])
\end{align*}
\]
Relational analysis: the lattice of polyhedra

- The value of the variables are represented by a polyhedron
- Lattice automata on $\Lambda = \Sigma \times \mathcal{P}$
- Abstract semantics:

$$\begin{align*}
\text{The lattice automaton is modified each time the value of } x \text{ changes}
\end{align*}$$
Conclusion and ongoing works

- Sumary of results
  - Approximate analysis of protocols using messages carrying values
  - Definition of a kind of automata dealing with infinite alphabets

- Ongoing work
  - Implementation of these algorithms
  - Experimentations
  - A new version of NBAC?