COMMUNICATING RECURSIVE PROGRAMS: CONTROL AND SPLIT-WIDTH

C. Aiswarya
Uppsala University, Sweden

Joint work with

Paul Gastin
LSV, ENS Cachan, France

K. Narayan Kumar
Chennai Mathematical Institute, India

ACTS
09/02/2015, Chennai
VERIFICATION

Model Checking

System $S$  Specification $\varphi$

$S \models \varphi$?

Refine $S$ (Fix bugs)

✓

✗
VERIFICATION

Model Checking

$S \models \varphi$?

Undecidable in many cases

System $S$
Specification $\varphi$

Refine $S$
(Fix bugs)
UNDER-APPROXIMATE VERIFICATION
UNDER-APPROXIMATE VERIFICATION
UNDER-APPROXIMATE VERIFICATION

Parametrised
UNDER-APPROXIMATE VERIFICATION

Parametrised
UNDER-APPROXIMATE VERIFICATION

Parametrised
UNDER-APPROXIMATE VERIFICATION

Parametrised
UNDER-APPROXIMATE VERIFICATION

- Parametrised
- Exhaustive
UNDER-APPROXIMATE VERIFICATION

- Parametrised
- Exhaustive
UNDER-APPROXIMATE VERIFICATION

Model Checking

System $S$
Specification $\varphi$

$S \models^k \varphi$?

> Decidable

Refine $S$
(Fix bugs)
UNDER-APPROXIMATE VERIFICATION

Model Checking

System $S$

Specification $\varphi$

Refine $S$

(Fix bugs)

Decidable

$S \models^k \varphi$?
UNDER-APPROXIMATE VERIFICATION

Model Checking

$S \models^k \varphi$?

System $S$
Specification $\varphi$

Refine $S$
(Fix bugs)

$\surd$ Decidable

$\times$
UNDER-APPROXIMATE VERIFICATION

Model Checking

System $S$
Specification $\phi$

Refine $S$
(Fix bugs)

$S \models^k \phi$?

Decidable

✗ ✓
UNDER-APPROXIMATE VERIFICATION

Model Checking

$S \models \phi$?

Decidable

System $S$

Specification $\varphi$

Refine $S$

(Fix bugs)
COMMUNICATING RECURSIVE PROGRAMS: CONTROL AND SPLIT-WIDTH
COMMUNICATING DISTRIBUTED SYSTEMS

Process 1

Network

Process 2

Process 3

Process 4
BEHAVIOURS:
MESSAGE SEQUENCE CHARTS

Proc 1
Proc 2
Proc 3
VERIFICATION PROBLEMS

- Emptiness or Reachability
- Inclusion or Universality
- Satisfiability $\phi$
- Model Checking: $S \models \phi$
- Temporal logics
- Propositional dynamic logics
- Monadic second order logic
COMMUNICATING RECURSIVE PROGRAMS:

- Turing powerful: verification undecidable
- Under-approximations
  - Decidable
  - Controllable
COMMUNICATING RECURSIVE PROGRAMS: CONTROL AND SPLIT-WIDTH

- Turing powerful: verification undecidable
- Under-approximations
  - Decidable
  - Controllable
CONTROLLERS FOR
VERIFICATION
OF COMMUNICATING SYSTEMS
COMMUNICATING DISTRIBUTED SYSTEMS

Process 1  Process 2  Process 3

Network

From  To
CONTROLLERS FOR DISTRIBUTED SYSTEMS

From

Controller 1

Process 1

Controller 2

Process 2

Controller 3

Process 3

To

Network
CONTROLLERS FOR DISTRIBUTED SYSTEMS

- Collection of local controllers
- Communication via piggy-backing
- Privacy: Do NOT read states/messages
LET’S DESIGN A CONTROLLER

UNDER-APPROXIMATION: BOUNDED (K) PHASE
PHASE

Receive from one process, send to all processes

Proc 1
Proc 2
Proc 3

time
PHASE

Receive from one process, send to all processes
Receive from one process, send to all processes
PHASE

Receive from one process, send to all processes
PHASE
Receive from one process, send to all processes

Proc 1
Proc 2
Proc 3
Receive from one process, send to all processes
PHASE

Receive from one process, send to all processes

Proc 1
Proc 2
Proc 3
**PHASE**
Receive from one process, send to all processes

**k-BOUNDED PHASE**
1. At most $k$ phases on each process
2. No cycles
**PHASE**
Receive from one process, send to all processes

**k-BOUNDED PHASE**
1. At most \( k \) phases on each process
2. No cycles
**PHASE**
Receive from one process, send to all processes

**k-BOUNDED PHASE**
1. At most $k$ phases on each process
2. No cycles
**PHASE**
Receive from one process, send to all processes

**k-BOUNDED PHASE**
1. At most $k$ phases on each process
2. No cycles
DISTRIBUTED CONTROLLER FOR K-BOUNDED PHASE U-A

A local controller for each process

Has a Phase Counter
Remembers current sender

Different sender? Increment counter; Update sender
Detect Cycle?
DISTRIBUTED CONTROLLER FOR K-BOUNDED PHASE U-A

Detect Cycle?

Phase Vectors

best info about phase number of other processes

Sends: tag with phase vector

Receives: update phase vector by taking MAX
CONTROLLERS FOR BOUNDED PHASE DISTRIBUTED SYSTEMS

- Collection of local controllers
- Communication via piggy-backing
- Privacy: Do NOT read states/messages

- System independent
- Generic
- Deterministic
- Finite state
DECIDABILITY OF
K BOUNDED PHASE
Polynomial SPLIT-WIDTH

- Decidable MSO
- Reachability
- Temporal Logics
- PSPACE
- PDL
Polynomial SPLIT-WIDTH

Refine phases to tree-like

bound split-width
ACYCLIC PHASE DECOMPOSITION
INDUCED GRAPH ON PHASE
INDUCED GRAPH ON PHASE
PHASE DECOMPOSITION
PHASE DECOMPOSITION

Tree-like
Polynomial SPLIT-WIDTH
Split-width

[Diagram of directed graph]

BUDGET
Diagram with vertices labeled as a, b, c, d, and edges connecting them with arrows.
Figure 4
A split decomposition of width 3.

Figure 5
A split term \( s \) (left) and a labelled term \( t \) (right) corresponding to Figure 4.

SPLIT TREE
OF THE FULL DECOMPOSITION
Figure 4
A split decomposition of width 3.

Figure 5
A split term $s$ (left) and a labelled term $t$ (right) corresponding to Figure 4.
Figure 4
A split decomposition of width 3.

Figure 5
A split term \(s\) (left) and a labelled term \(t\) (right) corresponding to Figure 4.

TREE INTERPRETATION
Split decomposition of width 3.

Tree Interpretation
Figure 4
A split decomposition of width 3.

TREE INTERPRETATION
Figure 4: A split decomposition of width 3.

Figure 5: A split term $s$ (left) and a labelled term $t$ (right) corresponding to Figure 4.
Figure 4: A split decomposition of width 3.

Figure 5: A split term (left) and a labelled term (right) corresponding to Figure 4.
Figure 4
A split decomposition of width 3.

Figure 5
A split term \(s\) (left) and a labelled term \(t\) (right) corresponding to Figure 4.

TREE INTERPRETATION
Figure 4
A split decomposition of width 3.

Figure 5
A split term (left) and a labelled term (right) corresponding to Figure 4.
Figure 4
A split decomposition of width 3.

Figure 5
A split term $s$ (left) and a labelled term $t$ (right) corresponding to Figure 4.
Figure 4
A split decomposition of width 3.

Figure 5
A split term (left) and a labelled term (right) corresponding to Figure 4.

TREE INTERPRETATION
Figure 4
A split decomposition of width 3.

Figure 5
A split term (left) and a labelled term (right) corresponding to Figure 4.

**TREE INTERPRETATION**
Figure 4

A split decomposition of width 3.

Figure 5

A split term \( s \) (left) and a labelled term \( t \) (right) corresponding to Figure 4.
Figure 4
A split decomposition of width 3.

Figure 5
A split term (left) and a labelled term (right) corresponding to Figure 4.
Figure 4: A split decomposition of width 3.

Figure 5: A split term $s$ (left) and a labelled term $t$ (right) corresponding to Figure 4.
Figure 4
A split decomposition of width 3.

Figure 5
A split term (left) and a labelled term (right) corresponding to Figure 4.
Figure 4
A split decomposition of width 3.

Figure 5
A split term (left) and a labelled term (right) corresponding to Figure 4.
Figure 4
A split decomposition of width 3.

Figure 5
A split term \(s\) (left) and a labelled term \(t\) (right) corresponding to Figure 4.

Tree Interpretation
Figure 4
A split decomposition of width 3.

Figure 5
A split term \(s\) (left) and a labelled term \(t\) (right) corresponding to Figure 4.

Tree interpretation
Figure 4
A split decomposition of width 3.

Figure 5
A split term (left) and a labelled term (right) corresponding to Figure 4.
Figure 4
A split decomposition of width 3.

Figure 5
A split term $s$ (left) and a labelled term $t$ (right) corresponding to Figure 4.

TREE INTERPRETATION
Figure 4
A split decomposition of width 3.

Figure 5
A split term \(s\) (left) and a labelled term \(t\) (right) corresponding to Figure 4.

Process edges
We refer to \([2, 15, 16]\) for more details and we summarise the computational complexities of

Then, the decision procedures can be restricted to the class

we have seen that existentially

can be recognised by a trivial 1-state

on the complexity of the decision procedures. We give below several examples.

reduces to the emptiness problem of

Another way to obtain

A

then the automaton

serves as

Focus now on the complexity of the decision procedures.

A tree automaton

Accepted by

A

from the

A

from


Finally, we deduce easily that

Now, let

be a sentence in

A

We have described above uniform decision procedures for an array of verification problems.

Verification procedures for other under-approximation classes.

with a further intersection

Characterising the under-approximation

Our approach is generic in

for

part of the input (in

Unary)

bound on split-width

bound on split-width

fixed

<table>
<thead>
<tr>
<th>Problem</th>
<th>Complexity</th>
</tr>
</thead>
<tbody>
<tr>
<td>CPDS emptiness</td>
<td>ExpTIME-Complete</td>
</tr>
<tr>
<td>CPDS inclusion or universality</td>
<td>2ExpTIME</td>
</tr>
<tr>
<td>CPDS inclusion or universality</td>
<td>ExpTIME-Complete</td>
</tr>
<tr>
<td>LTL / CPDL satisfiability or model checking</td>
<td>ExpTIME-Complete</td>
</tr>
<tr>
<td>ICPDL satisfiability or model checking</td>
<td>2ExpTIME-Complete</td>
</tr>
<tr>
<td>MSO satisfiability or model checking</td>
<td>Non-elementary</td>
</tr>
</tbody>
</table>
SPLIT-WIDTH 3
SPLIT-WIDTH 3
SPLIT-WIDTH 3
SPLIT WIDTH 3
SPLIT-WIDTH 3
SPLIT-WIDTH 3
SPLIT-WIDTH 3
SPLIT-WIDTH 3
UNDER-APPROXIMATE VERIFICATION

Model Checking

\[ S \models^k \varphi \]

System \( S \)

Specification \( \varphi \)

Decidable

Refine \( S \)

(Fix bugs)
OTHER UNDER-APPROXIMATIONS

* Bounded channel size
* Existentially bounded [Genest et al.]
* Acyclic Architectures [La Torre et al., Heußner et al. Clemente et al.]
* Bounded context switching [Qadeer, Rehof], [LaTorre et al.], ...
* Bounded phase [LaTorre et al.]
* Bounded scope [LaTorre et al.]
* Priority ordering [Atig et al., Saivasan et al.]
OTHER UNDER-APPROXIMATIONS

* Bounded channel size

* Existentially bounded [Genest et al.]

* Acyclic Architectures [La Torre et al., Heußner et al. Clemente et al.]

* Bounded context switching [Qadeer, Rehof], [LaTorre et al.], ...

* Bounded phase [LaTorre et al.]

* Bounded scope [LaTorre et al.]

* Priority ordering [Atig et al., Saivasan et al.]

Tree-width
OTHER UNDER-APPROXIMATIONS

* Bounded channel size

* Existentially bounded [Genest et al.]

* Acyclic Architectures [La Torre et al., Heußner et al. Clemente et al.]

* Bounded context switching [Qadeer, Rehof], [LaTorre et al.], ...

* Bounded phase [LaTorre et al.]

* Bounded scope [LaTorre et al.]

* Priority ordering [Atig et al., Saivasan et al.]

Tree-width

* Many of the above classes have bounded tree-width [Parlato, Madhusudhan]
OTHER UNDER-APPROXIMATIONS

- Acyclic Architectures (Constant)
- Bounded channel size (Bound + 2)
- Existentially bounded
- Bounded context switching (2^Bound)
- Bounded scope
- Bounded phase
- Priority ordering (Linear)
- Bounded Tree-width
Let $C$ be a class of bounded degree MSO definable graphs. TFAE
1. $C$ has a decidable MSO theory
2. $C$ can be interpreted in binary trees
3. $C$ has bounded tree-width
4. $C$ has bounded clique-width
5. $C$ has bounded split-width (for concurrent recursive behaviors)
Let $C$ be a class of bounded degree MSO definable graphs. TFAE

1. $C$ has a decidable MSO theory
2. $C$ can be interpreted in binary trees
3. $C$ has bounded tree-width
4. $C$ has bounded clique-width
5. $C$ has bounded split-width (for concurrent recursive behaviors)
Let $C$ be a class of bounded degree MSO definable graphs.

TFAE

1. $C$ has a decidable MSO theory
2. $C$ can be interpreted in binary trees
3. $C$ has bounded tree-width
4. $C$ has bounded clique-width
5. $C$ has bounded split-width (for concurrent recursive behaviors)
COMMUNICATING RECURSIVE PROGRAMS: CONTROL AND SPLIT-WIDTH
AUTONOMOUS COMPUTATIONS

• Recursive computations which does not read from other stacks/queues.
• A stretch of computation in which all incoming edges are on a single stack
• Recursive computations which does not read from other stacks/queues.

• A stretch of computation in which all incoming edges are on a single stack
PHASE

- A stretch of computation which reads from at most one stack/queue
PHASE

- A stretch of computation which reads from at most one stack/queue
- free (unlimited) autonomous computations
PHASE

- A stretch of computation which reads from at most one stack/queue
- free (unlimited) autonomous computations
- no loops
K-BOUNDED PHASE
K-BOUNDDED PHASE

Phase 1
Phase 2
Phase 3
IDENTIFYING AUTONOMOUS POPS

• Possible by tagging the values on stacks

• Deterministic controller for each stack

• The phase controller simulates one such automaton for each stack.
COMMUNICATING RECURSIVE PROGRAMS: CONTROL AND SPLIT-WIDTH


