Reasoning about Distributed Systems: WYSIWYG

Paul Gastin
LSV, ENS Cachan, France
C. Aiswarya
Uppsala University, Sweden
(p, on)(p, c₂!)(p, off)(q₂, c₂?)(p, c₁!)(q₁, c₁?)
(q₂, c₄!)(p, on)(p, c₂!)(p, off)(r, c₄?)(r, on)
(q₁, c₃!)(p, c₁!)(q₁, c₁?)(q₁, c₃!)(q₂, c₂?)(q₂, c₄!)
(r, c₄?)(r, on)(r, c₃?)(r, off) · · ·
(p, on)(p, c₂!)(p, off)(q₂, c₂?)(p, c₁!)(q₁, c₁?)
(q₂, c₄!)(p, on)(p, c₂!)(p, off)(r, c₄?)(r, on)
(q₁, c₃!)(p, c₁!)(q₁, c₁?)(q₁, c₃!)(q₂, c₂?)(q₂, c₄!)
(r, c₄?)(r, on)(r, c₃?)(r, off) …
System: Concurrent Processes with Data-Structures

- Processes
- Data structures
  - Stacks: recursive programs, multithreaded
  - Queues: communication (FIFO)
  - Bags: communication (unordered)
Architectures: Special cases

- PDA: Pushdown automata
- Recursive programs
Architectures: Special cases

- PDA: Pushdown automata
  Recursive programs
- MPDA: Multi-pushdown automata
  Multi-threaded recursive programs
Architectures: Special cases

• PDA: Pushdown automata
  Recursive programs

• MPDA: Multi-pushdown automata
  Multi-threaded recursive programs

• MPA: Message passing automata
  Communicating finite state machines
Architectures: Special cases

- PDA: Pushdown automata
  Recursive programs
- MPDA: Multi-pushdown automata
  Multi-threaded recursive programs
- MPA: Message passing automata
  Communicating finite state machines
- PN: Petri Nets
  Only bags
Remote on-off via 2 channels

Diagram showing a network with nodes labeled p, q_1, q_2, r, and channel labels c_1, c_2, c_3, c_4.
System: Architecture + Boolean Programs

Diagram depicting system architecture with Boolean programs.
Operational semantics

* Transition system TS
* States (infinite)
  * locations of processes
  * contents of data structures
* Transitions
  * Induced by the boolean programs
* Linear traces: abstractions of runs of TS
Linear Traces

\[(p, \text{on})(p, c_2!)(p, \text{off})(q_1, c_1?)(q_1, c_1?)(p, \text{on})(p, c_2!)(p, \text{off})(r, c_4?)(r, \text{on})(q_1, c_3!)(p, c_1!)(q_1, c_1?)(q_1, c_3!)(q_2, c_2?)(q_2, c_4!)(r, c_4?)(r, \text{on})(r, c_3?)(r, \text{off})\ldots\]
Linear Traces vs. Graphs

\[(p, \text{on})(p, c_2!)(p, \text{off})(q_2, c_2?)(p, c_1!)(q_1, c_1?)\]
\[(q_2, c_4!)(p, \text{on})(p, c_2!)(p, \text{off})(r, c_4?)(r, \text{on})\]
\[(q_1, c_3!)(p, c_1!)(q_1, c_1?)(q_1, c_3!)(q_2, c_2?)(q_2, c_4!)\]
\[(r, c_4?)(r, \text{on})(r, c_3?)(r, \text{off})\ldots\]
Obey the Latest Order

\((p, \text{on})(p, c_2!)(p, \text{off})(q_2, c_2?)(p, c_1!)(q_1, c_1?)\)
\((q_2, c_4!)(p, \text{on})(p, c_2!)(p, \text{off})(r, c_4?)(r, \text{on})\)
\((q_1, c_3!)(p, c_1!)(q_1, c_1?)(q_1, c_3!)(q_2, c_2?)(q_2, c_4!)\)
\((r, c_4?)(r, \text{on})(r, c_3?)(r, \text{off})\cdots\)
Answer the correct client for topmost requests

WYSIWYG: Make visible what is important
Graphs for Sequential Systems

Answer the correct client for topmost requests

Behaviors should be graphs

Make visible what is important

Nested Words
Alur, Madhusudan, 2009
Semantics of CPDS on Graphs
Semantics of CPDS on Graphs
Semantics of CPDS on Graphs
Semantics of CPDS on Graphs
Outline

- Concurrent Processes with Data Structures
- Behaviors as Graphs
  - Specifications
  - Verification with Graphs and under-approximations
  - Split-width and tree interpretation
  - Conclusion
Specification over Graphs

MSO: Monadic Second Order Logic

\[ \varphi ::= \text{false} \mid a(x) \mid p(x) \mid x \leq y \mid x \triangleright^d y \mid x \rightarrow y \]
\[ \mid x \in X \mid \varphi \lor \varphi \mid \neg \varphi \mid \exists x \, \varphi \mid \exists X \, \varphi \]
Specification over Graphs
Obey the latest order

TL

\[ G(r \land on \Rightarrow \text{Latest}_p Y_p on) \]

FO

\[ \forall z \ (r(z) \land on(z)) \Rightarrow \exists y \ (p(y) \land y < z) \land \forall x \ (x < z \land p(x) \Rightarrow x \leq y) \land \exists x \ (x \rightarrow y \land on(x)) \]
Specification over Linear Traces

\((p, \text{on})(p, c_2!)(p, \text{off})(q_2, c_2?)(p, c_1!)(q_1, c_1?)\)

\((q_2, c_4!)(p, \text{on})(p, c_2!)(p, \text{off})(r, c_4?)(r, \text{on})\)

\((q_1, c_3!)(p, c_1!)(q_1, c_1?)(q_1, c_3!)(q_2, c_2?)(q_2, c_4!)(r, c_4?)(r, \text{on})\)

\((r, c_4?)(r, \text{on})(r, c_3?)(r, \text{off})\) \cdots

* Based on the word successor relation, and the word total order

* LTL over words, MSO over words

* LTL specification are not always meaningful

LTL \(\setminus X\), Closure properties, ...

* Natural properties of graphs are difficult or impossible to express
  on linear traces
Based on the word successor relation, and the word total order, LTL over words, MSO over words

LTL specification are not always meaningful
LTL \ X, Closure properties, ...

Natural properties of graphs are difficult or impossible to express on linear traces
Graphs for Sequential Systems

\[ \forall x, y \left( a(x - 1) \land x \triangleright y \land \neg \exists z, z' (z \triangleright z' \land z < x < z') \right) \Rightarrow a(y + 1) \]
Answer the correct client for topmost requests

Specifications should be on graphs

Not expressible in MSO over Linear Traces without nesting relation even with visible alphabet
Concurrent Processes with Data Structures
Behaviors as Graphs
Specifications

* Verification with Graphs and under-approximations
* Split-width and tree interpretation
* Conclusion
Verification problems

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

$\forall z \ (r(z) \land \text{on}(z)) \Rightarrow \exists y \ (p(y) \land y < z)$

$\land \forall x \ (x < z \land p(x) \Rightarrow x \leq y)$

$\land \exists x \ (x \rightarrow y \land \text{on}(x))$
Model Checking vs Reachability

* Reachability reduces to model checking
* Model checking reduces to Reachability ...
  ... when specifications can be translated to automata
  ... this is not possible in general for graphs
Verification problems

* Emptiness or Reachability
* Inclusion or Universality
* Satisfiability $\phi$
* Model checking $S \models \phi$
* Temporal logics
  - Propositional dynamic logics
  - Monadic second order logic

Temporal logic example:

\[
G(r \land on \Rightarrow \text{Latest}_p Y_p on)
\]

\[
\forall z \ (r(z) \land on(z)) \Rightarrow \exists y \ (p(y) \land y < z)
\]

\[
\land \forall x \ (x < z \land p(x) \Rightarrow x \leq y)
\]

\[
\land \exists x \ (x \to y \land on(x))
\]

Undecidable in general
Mainly for reachability

* Emptiness or Reachability
* Inclusion or Universality

$\models\phi$

Temporal logics

* Propositional dynamic logics
* Monadic second order logic

Undecidable

* Bounded data structures
* 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.]
Under-approximate Verification

Model checking problem: \( S \models_C \phi \)

\( C: \) class of behaviors

\( \phi: \) Specification

\( \phi_s \Rightarrow \phi \) is valid in \( C \)
Graph Structure and Monadic Second-Order Logic
A Language-Theoretic Approach

BRUNO COURCELLE
Université de Bordeaux

JOOST ENGELFRIET
Universiteit Leiden
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 CBMs)

Decidability of MSO theory

Reduction to the theory of Tree Automata
The Tree Width of Auxiliary Storage

P. Madhusudan
University of Illinois at Urbana-Champaign, USA
madhu@illinois.edu

Gennaro Parlato
LIAFA, CNRS and University of Paris Diderot, France.
gennaro@liafa.jussieu.fr

Abstract

We propose a generalization of results on the decidability of emptiness for several restricted classes of sequential and distributed automata with auxiliary storage (stacks, queues) that have recently been proved. Our generalization relies on reducing emptiness of these automata to finite-state graph automata (without storage) restricted to monadic second-order (MSO) definable graphs of bounded tree-width, where the graph structure encodes the mechanism of the storage. Intuitively, a symbol that gets stored or multiple directed paths augmented with special edges to capture the retrieval point by using an appropriate tiling of this special edge.

However, the various identified decidable restrictions on the automata are, for the most part, awkward in their definitions. e.g. emptiness of multi-stack pushdown automata where push to any stack is allowed at any time, but popping is restricted to the first non-empty stack is decidable! [8]. Yet, relaxing the definitions to more natural ones seems to either destroy decidability or their power. It is hence natural to ask: why do these automata have decidable emptiness problems? Is there a common underlying principle for these classes in decidability? 
Concurrent Processes with Data Structures

Behaviors as Graphs

Specifications

Verification with Graphs and under-approximations

* Split-width and tree interpretation

* Conclusion

joint work with

C. Aiswarya

K. Narayan Kumar
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 CBMs)
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 CBMs)

$k \leq 120(t + 1)$

$k \leq 2c - 3$
SPLIT DECOMPOSITION OF CBMs
SPLIT DECOMPOSITION OF CBMs
SPLIT DECOMPOSITION OF CBMs
SPLIT DECOMPOSITION OF CBMs
SPLIT DECOMPOSITION OF CBMs
SPLIT DECOMPOSITION OF CBMs
SPLIT DECOMPOSITION OF CBMs
SPLIT DECOMPOSITION OF CBMs
SPLIT DECOMPOSITION OF CBMs
SPLIT DECOMPOSITION OF CBMs
SPLIT DECOMPOSITION OF CBMs
SPLIT DECOMPOSITION OF CBMs
SPLIT DECOMPOSITION OF CBMs
SPLIT DECOMPOSITION OF CBMs
SPLIT DECOMPOSITION OF CBMs
SPLIT DECOMPOSITION OF CBMs
SPLIT DECOMPOSITION OF CBMs
SPLIT DECOMPOSITION OF CBMs
SPLIT DECOMPOSITION OF CBMs
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
Tree interpretation in
Abstract Tree 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.

Vertices are leaves
Data edges

Tree interpretation in Abstract Tree Decomposition
Tree interpretation in Abstract Tree Decomposition

Process edges

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.
Nested Words: split-width $\leq 2$

$$w ::= \bullet \mid w \rightarrow w \mid \quad \overset{\text{blue}}{ullet \rightarrow w \rightarrow \bullet} \mid \quad \overset{\text{blue}}{\bullet \rightarrow}$$
Split-width: under-approximations

- Words
- Nested Words
- Acyclic Architectures

- Bounded channel size
- Existentially bounded
- Bounded context switching
- Bounded scope

- Bounded phase
- Priority ordering

Constant
Bound + 2
$2^{\text{Bound}}$
### Split-width: parametrized verification

<table>
<thead>
<tr>
<th>Problem</th>
<th>Complexity</th>
<th>Complexity</th>
</tr>
</thead>
<tbody>
<tr>
<td>CPDS emptiness</td>
<td>ExpTime-Complete</td>
<td>PTIME-Complete</td>
</tr>
<tr>
<td>CPDS inclusion or universality</td>
<td>2ExpTime</td>
<td>ExpTime-Complete</td>
</tr>
<tr>
<td>LTL / CPDL satisfiability or model checking</td>
<td>ExpTime-Complete</td>
<td>ExpTime-Complete</td>
</tr>
<tr>
<td>ICPDL satisfiability or model checking</td>
<td>2ExpTime -Complete</td>
<td>Non-elementary</td>
</tr>
<tr>
<td>MSO satisfiability or model checking</td>
<td>Non-elementary</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Problem</th>
<th>Complexity</th>
<th>Complexity</th>
</tr>
</thead>
<tbody>
<tr>
<td>CPDS emptiness</td>
<td>ExpTime-Complete</td>
<td>PTIME-Complete</td>
</tr>
<tr>
<td>CPDS inclusion or universality</td>
<td>2ExpTime</td>
<td>ExpTime-Complete</td>
</tr>
<tr>
<td>LTL / CPDL satisfiability or model checking</td>
<td>ExpTime-Complete</td>
<td>ExpTime-Complete</td>
</tr>
<tr>
<td>ICPDL satisfiability or model checking</td>
<td>2ExpTime -Complete</td>
<td>Non-elementary</td>
</tr>
<tr>
<td>MSO satisfiability or model checking</td>
<td>Non-elementary</td>
<td></td>
</tr>
</tbody>
</table>

C. Aiswarya, P.G, K. Narayan Kumar
Concurrent Processes with Data Structures
Behaviors as Graphs
Specifications
Verification with Graphs and under-approximations
Split-width

Conclusion
# Understanding Behaviors

<table>
<thead>
<tr>
<th>Linear Traces</th>
<th>Graphs (CBMs)</th>
</tr>
</thead>
</table>
| • Interleaved sequence of events. **Interactions are obfuscated** and very difficult to recover. | • Visual description of behavior  
• Interactions are visible  
• no combinatorial explosion                                                   |
| • Successor relation not meaningful                                             |                                                                                                     |
| • Combinatorial explosion  
single distributed behavior results in a huge number of linear traces    |                                                                                                     |
## WYSIWYG

### Expressiveness of Specifications

<table>
<thead>
<tr>
<th>Linear Traces</th>
<th>Graphs (CBMs)</th>
</tr>
</thead>
</table>
| • Too weak for many natural specifications  
• Requires syntactical or semantical restrictions to be meaningful | • Powerful specifications  
• Interactions are built-in  
• Meaningful |

---

The table above compares Linear Traces and Graphs (CBMs) in terms of expressiveness of specifications. Linear Traces are too weak for many natural specifications and require syntactical or semantical restrictions to be meaningful. Graphs (CBMs), on the other hand, are powerful specifications that include built-in interactions and are meaningful.
### Efficiency of Algorithms

<table>
<thead>
<tr>
<th>Linear Traces</th>
<th>Graphs (CBMs)</th>
</tr>
</thead>
<tbody>
<tr>
<td>• Undecidable in general</td>
<td>• Undecidable in general</td>
</tr>
<tr>
<td>• Decidable under restrictions</td>
<td>• Decidable under more lenient restrictions</td>
</tr>
<tr>
<td>• Reductions to word automata</td>
<td>• Reductions to tree automata via tree-interpretations</td>
</tr>
<tr>
<td>• Good space complexity</td>
<td>• Good time complexity</td>
</tr>
<tr>
<td>• Many tools available</td>
<td>• Tools to be developed</td>
</tr>
</tbody>
</table>

**WYSIWYG**
Conclusion

- Use graphs to reason about behaviors of systems distributed or sequential
- Exploit graph theory
  Logics, decompositions, tree interpretations
- Split-width: convenient decomposition technique
  as powerful as tree-width or clique-width for CBMs
  yields optimal algorithms
Perspectives

* Extensions
  * Timed systems
  * Dynamic creation of processes
  * Read from many
  * Infinite behaviors
  *...

* Tools

THANK YOU