Reasoning about Distributed Systems: WYSIWYG

Paul Gastin
LSV, ENS Cachan, France
C. Aiswarya
CMI, Chennai, India

Proceedings of FSTTCS’14 (Invited talk)
Introduction

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

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

Obey the latest order
Operational semantics

* Transition system TS

* Configurations (infinite)
  * local states of processes
  * contents of data structures

* Transitions
  * Induced by the boolean programs

* Linear traces: abstractions of runs of TS
Linear Traces

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

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

Does it obey the latest order?
Concurrent Processes with Data Structures

* Behaviors as Graphs
* Specifications
* Verification with Graphs and under-approximations
* Split-width and tree interpretation
* Conclusion
Linear Traces

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

Does it obey the latest order?
(p, on)(p, c_2!)(p, off)(r, c_4?) (q_1, c_1?) (q_2, c_4!)(p, on)(p, c_2!)(p, off)(r, c_4?) (r, 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, on)(r, c_3?)(r, off) \ldots

Does it obey the latest order?
Linear Traces vs. Graphs

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

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

$(q_1, c_3!)(p, c_1!)(q_1, c_1?)(q_1, c_3!)(q_2, c_2?)(q_2, c_4!)$

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

Message Sequence Charts
ITU Standard
Linear Traces vs. Graphs

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

Does it obey the latest order?
Does it obey the latest order?

Linear Traces vs. Graphs
Graphs for Sequential Systems

Answer the correct client for topmost requests

WYSIWYG: Make visible what is important
Graphs for Sequential Systems

Answer the correct client for topmost requests

WYSIWYG: Make visible what is important

Nested Words
Alur, Madhusudan, 2009
## Understanding Behaviors

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

**WYSIWYG**
## WYSIWYG

### Understanding Behaviors

<table>
<thead>
<tr>
<th>Linear Traces</th>
<th>Graphs (CBMs)</th>
</tr>
</thead>
<tbody>
<tr>
<td>• Interleaved sequence of events. Interaction are obfuscated and very difficult to recover.</td>
<td>Behaviors should be graphs. Interaction sequences are visible behaviors are not combinatorial explosion.</td>
</tr>
<tr>
<td>• Successor relation not meaningful</td>
<td></td>
</tr>
<tr>
<td>• Combinatorial explosion single distributed behavior results in a huge number of linear traces</td>
<td></td>
</tr>
</tbody>
</table>
Semantics of CPDS on Graphs

\[ a \xrightarrow{a} b \xrightarrow{c} d \]

\[ b \xrightarrow{a} c \xrightarrow{d} c \]
Semantics of CPDS on Graphs

\begin{figure}
\centering
\includegraphics[width=\textwidth]{figure}
\caption{Diagram of CPDS on graphs showing transitions and states for a, b, c, and d.}
\end{figure}
Semantics of CPDS on Graphs
Semantics of CPDS on Graphs
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

\( \varphi ::= \text{false} \mid a(x) \mid p(x) \mid x \leq y \mid x \triangleright^d y \mid x \rightarrow y \)

\[ x \in X \mid \varphi \lor \varphi \mid \neg \varphi \mid \exists x \, \varphi \mid \exists X \, \varphi \]
Specification over Graphs
Obey the latest order

TL

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

FO

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

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

\[ \land \exists x (x \rightarrow y \land \text{on}(x)) \]
**Specification over Linear Traces**

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

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

* LTL over words, MSO over words

---

Process successor can be recovered

Data edges cannot in general
Based on the word successor relation, and the word total order
LTL over words, MSO over words

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

Obey the latest order
not expressible
in MSO over Linear Traces

* Based on word total order
* LTL over words, MSO over words

Process successor can be recovered
Data edges cannot in general
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
Answer the correct client for topmost requests

∀x, y (a(x − 1) ∧ x ▷ y ∧ ¬∃z, z' (z ▷ z' ∧ z < x < z')) ⇒ a(y + 1)
Graphs for Sequential Systems

Answer the correct client for topmost requests

Not expressible in MSO over Linear Traces without nesting relation even with visible alphabet
## WYSIWYG

### Expressiveness of Specifications

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

**CBBs**
WYSIWYG
Expressiveness of Specifications

<table>
<thead>
<tr>
<th>Linear Traces</th>
<th>Graphs (CBMs)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Too weak for many natural specifications</td>
<td><strong>Specifications should be on graphs</strong></td>
</tr>
<tr>
<td>Difficult to write/understand</td>
<td>Powerful specifications</td>
</tr>
<tr>
<td>Requires syntactical or semantical restrictions</td>
<td>Meanings are built-in</td>
</tr>
<tr>
<td>Easy to write/understand</td>
<td>• Easy to write/understand</td>
</tr>
</tbody>
</table>

Specifications should be on graphs
Outline

☑ 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

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

$$\forall z (r(z) \land \text{on}(z)) \Rightarrow \exists y (p(y) \land y < z)$$
$$\land \forall x (x < z \land p(x) \Rightarrow x \leq y)$$
$$\land \exists x (x \rightarrow y \land \text{on}(x))$$
Verification problems

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

Undecidable in general

Obey the latest order

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

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

$\land \exists x (x \rightarrow y \land \text{on}(x))$
Under-approximate Verification

Satisfiability problem:

Is $\phi$ satisfiable in $C$?

$C$: class of behaviors

$\phi$: Specification
Under-approximate Verification

Emptiness or reachability problem:

Is there a run of $S$ on some behavior from $C$?

C: class of behaviors

S: CPDS
Under-approximate Verification

Model checking problem: \( S \models_{\mathcal{C}} \phi \)

Do all behaviors from \( \mathcal{C} \) accepted by \( S \) satisfy \( \phi \)?

C: class of behaviors

\( \phi \): Specification

S: CPDS
Verification problems

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

Obey the latest order

Undecidable in general
Under-approximate Verification

Mainly for reachability

* Emptiness or Reachability
* Inclusion or Universality
* Satisfiability

$\mathcal{S} \models \phi$

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.]
Model Checking vs Reachability

* Reachability reduces to model checking

* Model checking reduces to Reachability ...

... when specifications can be translated to systems

... this is not possible in general for graphs

\[
S \models \phi \\
S \cap S_{\neg \phi} = \emptyset
\]
Model Checking vs Reachability

* Reachability reduces to model checking

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

\[ S \models \phi \]

\[ S \cap S_{\neg \phi} = \emptyset \]
Under-approximate Verification

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

- **C**: class of behaviors
- **\( \phi \)**: Specification
- **\( S \)**: CPDS

\( S \models_C \phi \) iff \( \phi_S \Rightarrow \phi \) 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)

Reduction to the theory of Tree Automata
Under-approximate Verification

- Emptiness or Reachability
- Inclusion or Universality

\( S \models \phi \)

Temporal logics
- Propositional dynamic logics
- Monadic second order logic

Mainly for reachability

- 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.]


**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 at some point has to be retrieved at some future point by using an appropriate tiling of this special edge. Between the point where the symbol gets stored to the point where it is retrieved, the mechanism of the storage is captured using states.

However, the various identified decidable restrictions on these automata are, for the most part, awkward in their definitions—e.g., emptiness of multi-stack pushdown automata where pushing 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 that explains their decidability?
Outline

- 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)

\[
t \leq 2(k + |\text{Procs}|) - 1
\]
\[
c \leq 2(k + |\text{Procs}|) + 1
\]
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)

\[ k \leq 120(t + 1) \]

\[ k \leq 2c - 3 \]
Decomposition game

- Eve: Disconnect the graph by cutting edges
- Adam: Choose a connected component
- Split game: Eve cuts process edges only (CBM)
- Width: Maximal number of holes in graphs (CBMs) along a play until reaching an atomic graph
SPLIT DECOMPOSITION OF CBMs
SPLIT DECOMPOSITION OF CBMs

3 splits
SPLIT DECOMPOSITION OF CBMs
SPLIT DECOMPOSITION OF CBMs

1 hole left
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

3 holes
SPLIT DECOMPOSITION OF CBMs
SPLIT DECOMPOSITION OF CBMs
SPLIT DECOMPOSITION OF CBMs
SPLIT DECOMPOSITION OF CBMs

2 holes
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

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.

Data edges

Tree interpretation in
Abstract Tree Decomposition
Data edges

Tree interpretation in
Abstract Tree Decomposition
Process edges

Tree interpretation in
Abstract Tree Decomposition
Tree interpretation in Abstract Tree Decomposition
Tree interpretation in Abstract Tree Decomposition
Process edges

Tree interpretation in
Abstract Tree Decomposition
Process edges

Tree interpretation in
Abstract Tree Decomposition
Process edges

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.

Process edges

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

Process edges

successor?
Process edges

Tree interpretation in Abstract Tree Decomposition
In the abstract tree, we can interpret the graph (CBM)
• vertices and labels
• data edges
• process edges
with tree (walking) automata

PDL or MSO formulas
Split-width: under-approximations

- Words
- Nested Words
- Mazurkiewicz Traces
- Acyclic Architectures
- Bounded channel size
- Existentially bounded
- Bounded context switching
- Bounded scope
- Bounded phase
- Priority ordering

- Constant
- \(2^{\text{Bound}}\)
reachability/emptiness
reachability/emptiness
reachability/emptiness
reachability/emptiness
reachability/emptiness
reachability/emptiness
## Split-width: parametrized verification

<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>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>

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>
<tbody>
<tr>
<td>• Interleaved sequence of events. <strong>Interactions are obfuscated</strong> and very difficult to recover.</td>
<td></td>
</tr>
<tr>
<td>• <strong>Successor relation not meaningful</strong></td>
<td>• <strong>Visual description of behavior</strong></td>
</tr>
<tr>
<td>• <strong>Combinatorial explosion</strong></td>
<td>• <strong>Interactions are visible</strong></td>
</tr>
<tr>
<td>single distributed behavior results in a huge number of linear traces</td>
<td>• <strong>no combinatorial explosion</strong></td>
</tr>
</tbody>
</table>
# Expressiveness of Specifications

<table>
<thead>
<tr>
<th>Linear Traces</th>
<th>Graphs (CBMs)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td>• Too weak for many natural specifications</td>
<td>• Powerful specifications</td>
</tr>
<tr>
<td>• Difficult to write/understand</td>
<td>• Interactions are built-in</td>
</tr>
<tr>
<td>• Requires syntactical or semantical restrictions to be meaningful</td>
<td>• Meaningful</td>
</tr>
<tr>
<td></td>
<td>• Easy to write/understand</td>
</tr>
</tbody>
</table>
### 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>
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
  * Parameterized systems (size, topology)
    with Marie Fortin, FOSSACS’16
  * Timed systems
    with S. Akshay and S. Krishna, submitted
  * Higher-order PDA
    with C. Aiswarya and P. Saivasan
  * Dynamic creation of processes
  * Read from many
  * Infinite behaviors
  * ...

* Tools
Perspectives

* Extensions
  * Parameterized systems (size, topology)
    with Marie Fortin, FOSSACS’16
  * Timed systems
    with S. Akshay and S. Krishna, submitted
  * Higher-order PDA
    with C. Aiswarya and P. Saivasan
  * Dynamic creation of processes
  * Read from many
  * Infinite behaviors
  * ...

* Tools

THANK YOU