Verification of Run Graphs
(Some Preliminary Results)

Alexander Heußner
LaBRI Bordeaux, France
A distributed system consists of a collection of distinct processes which are spatially separated and communicate with one another by exchanging messages.

- e.g., Apps based on Berkeley Socket API / MPI
- model for asynchronous multiprocessors (buzzword: (S)inglechip(C)loud(C)omputer)
- include other ways of synchronization
Distributed Systems

“A distributed system consists of a collection of distinct processes which are spatially separated & communicate with one another by exchanging messages.”

- e.g., Apps based on Berkeley Socket API / MPI
- model for asynchronous multiprocessors (buzzword: (S)inglechip(C)loud(C)omputer)
- include other ways of synchronization
Queueing Communicating Processes

- Processes modeled as local labeled transition system with "(infinite) data"
  - pushdown stack
  - variables
  - ...

- Add synchronization via
  - Reliable, unbounded, fifo queues
  - Locks for shared resources
  - (Classical) rendez-vous
  - Barriers
  - ...

(Q)ueueing (C)ommunicating (P)rocesses
Recap: Ongoing Research

- **fifo queues (I):** Message Sequence Charts \textbf{MSC}
  - bounded branching of unfolding [Madhusudan ’01]
  - \(\exists/\forall\) bounded channels [Lohrey/Musholl ’04]
  - ...

- **fifo queues (II)** [La Torre/Madhusudan/Parlato ’08]
  - assert channels can only receive when local stack empty
  - communication architecture is a tree

- **locks / monitors** [Kahlon/Gupta/…’07–’10]

- **fifo queues (III)** [Heußner/Leroux/Muscholl/Sutre ’10]
  - generalize previous stack-queue-interplay restriction
  - focus atomic send-receives (semantic restriction)
  - communication architecture does not allow
Recap: Ongoing Research

- **fifo queues (I):** Message Sequence Charts **MSC**
  - bounded branching of unfolding [Madhusudan '01]
  - $\exists/\forall$ bounded channels [Lohrey/Musholl '04]
  - ...

- **fifo queues (II)** [La Torre/Madhusudan/Parlato '08]
  - assert channels can only receive when local stack empty
  - communication architecture is a tree

- **locks / monitors** [Kahlon/Gupta/... '07–'10]

- **fifo queues (III)** [Heußner/Leroux/Muscholl/Sutre '10]
  - generalize previous stack-queue-interplay restriction
  - focus atomic send-receives (semantic restriction)
  - communication architecture does not allow semantic restrictions!
Recap: Ongoing Research

- **fifo queues (I)**: Message Sequence Charts MSC
  - bounded branching of unfolding [Madhusudan '01]
  - $\exists / \forall$ bounded channels [Lohrey/Musholl '04]
  - ...

- **fifo queues (II)** [La Torre/Madhusudan/Parlato '08]
  - assert channels can only receive when local stack empty
  - communication architecture is a tree

- **locks / monitors** [Kahlon/Gupta/... '07–'10]

- **fifo queues (III)** [Heußner/Leroux/Muscholl/Sutre '10]
  - generalize previous stack-queue-interplay restriction
  - focus atomic send-receives (semantic restriction)
  - communication architecture does not allow
Recap: Ongoing Research

- **fifo queues (I): Message Sequence Charts MSC**
  - bounded branching of unfolding [Madhusudan '01]
  - $\exists/\forall$ bounded channels [Lohrey/Muscholl '04]
  - ...

- **fifo queues (II) [La Torre/Madhusudan/Parlato '08]**
  - assert channels can only receive when local stack empty
  - communication architecture is a tree

- **locks / monitors [Kahlon/Gupta/... '07–'10]**

- **fifo queues (III) [Heußner/Leroux/Muscholl/Sutre '10]**
  - generalize previous stack-queue-interplay restriction
  - focus atomic send-receives (semantic restriction)
  - communication architecture does not allow semantic restrictions!
Antipatterns/“Contra”-Patterns

Distinctive causal entanglement patterns lead to undecidability issues

[Brand/Zafiropoulo ’83]

Multi-stack PDA
Decidability/“Pro”-Patterns

- cut run into sequence of sub-patterns
- only “exchange” finite information between these patterns
- sub-patterns either use recursion or composition of sub-sub-patterns
- “context-freeness”
How to Bridge The Gap?
Run Graphs

- each process defines a local total order of events
- additional causal constraints: synchronization (fifo), local data (push/pop)

hence, partial order semantics!

1 aka “space-time diagrams”, extended Hasse diagrams, MSC with pushdowns, ...
model checking MSO on QCP

model checking MSO on RunGraphs

... is decidable

... is decidable

“Easier” & more Concise Question, but still...
Model checking MSO on QCP is decidable.

Model checking MSO on RunGraphs is decidable.

Most Important Piece!

Graph has finite treewidth iff MSO on Graphs is decidable... is decidable... is decidable.
Standing on the Shoulder of Giant Results

- **treewidth** measures how close a graph is to a tree
- classical way to calculate: tree decomposition
  \[ \text{treewidth} = \max \text{ size of "bag" } - 1 \]
- apply (some) tree-based algos to general graphs via \( TD_G \)
- extension to classes of graphs leads to unbounded treewidth

[Courcelle’s Theorem]
SAT(MSO) is fixed parameter tractable for given treewidth of model and size of formula.

[Madhusudan/Parlato ’10(?)] based on [Seese ’91]
Given class \( C \) of graphs of bounded treewidth which are MSO definable, and an MSO formula \( \phi \), then we can test whether there exists \( G \in C \) s.t. \( G \models \phi \).
Standing on the Shoulder of Giant Results

- **treewidth** measures how close a graph is to a tree
- classical way to calculate: tree decomposition
- apply (some) tree-based algs to general graphs via $TD_G$
- extension to classes of graphs leads to unbounded treewidth

[Courcelle’s Theorem]
SAT(MSO) is fixed parameter tractable for given treewidth of model and size of formula.

[Madhusudan/Parlato ’10(?)] based on [Seese ’91]
Given class $C$ of graphs of bounded treewidth which are MSO definable, and an MSO formula $\varphi$, then we can test whether there exists $G \in C$ s.t. $G \models \varphi$. 
Standing on the Shoulder of Giant Results

- treewidth measures how close a graph is to a tree
- classical way to calculate: tree decomposition

\[ \text{treewidth} = \text{max size of "bag" } - 1 \]

- apply (some) tree-based algos to general graphs via \( TD_G \)
- extension to classes of graphs leads to unbounded treewidth

[Courcelle’s Theorem]
SAT(MSO) is fixed parameter tractable for given treewidth of model and size of formula.

[Madhusudan/Parlato ’10(?)]
based on [Seese ’91]

Given class \( C \) of graphs of bounded treewidth which are MSO definable, and an MSO formula \( \varphi \), then we can test whether there exists \( G \in C \) s.t. \( G \models \varphi \).
Standing on the Shoulder of Giant Results

- **treewidth** measures how close a graph is to a tree
- classical way to calculate: tree decomposition
  
  \[ TD_G \]
  
  treewidth = max size of “bag” – 1

- apply (some) tree-based algos to general graphs via \( TD_G \)
- extension to classes of graphs leads to **unbounded treewidth**

[Courcelle’s Theorem]
SAT(MSO) is fixed parameter tractable for given treewidth of model and size of formula.

[Madhusudan/Parlato ’10(?)] based on [Seese ’91]
Given class \( C \) of graphs of bounded treewidth which are MSO definable, and an MSO formula \( \varphi \), then we can test whether there exists \( G \in C \) s.t. \( G \models \varphi \).
model checking MSO on QCP

... is decidable

model checking MSO on RunGraphs

... is decidable

Graph has finite treewidth

MSO on Graphs is decidable

Gennaro’s Piece [Parlato & Madhusudan]
model checking MSO on QCP is decidable

model checking MSO on RunGraphs is decidable

Graph has finite treewidth

MSO on Graphs is decidable

Alex’s (Alternative) Piece
Hyperedge Replacement Grammars

∀ Hypergraph \( \mathcal{H} = \langle V, E, Ext \rangle \)
  - vertices \( V \), edges \( E \in V^+ \)
  - external vertices \( Ext \subseteq V \)

∀ HR-grammar \( G = \langle N, T, R, n^\bullet \rangle \)
  - non-/terminals \( N/T \)
  - initial \( n^\bullet \in N \)
  - rules \( R \)

∀ rule \( R : X \in N \rightarrow H \)
  - \( \mathcal{H} \) is hypergraph whose
    (i) vertices are from \( N \cup T \)
    (ii) \( X \) has “arity” \( |Ext_\mathcal{H}| \)

∀ rule-width of \( G \):
  \( |\{\text{vertices of rule’s rhs}\}| - 1 \)
Hyperedge Replacement Grammars

- Hypergraph $\mathcal{H} = \langle V, E, Ext \rangle$
  - vertices $V$, edges $E \in V^+$
  - external vertices $Ext \subseteq V$

- HR-grammar $G = \langle N, T, R, n^\bullet \rangle$
  - non-/terminals $N/T$
  - initial $n^\bullet \in N$
  - rules $R$

- Rule $R : X \in N \rightarrow \mathcal{H}$
  - $\mathcal{H}$ is hypergraph whose
    - (i) vertices are from $N \cup T$
    - (ii) $X$ has “arity” $|Ext_\mathcal{H}|$

- Rule-width of $G$:
  $|\{\text{vertices of rule's rhs}\}| - 1$
Hyperedge Replacement Grammars

 Hypergraph $\mathcal{H} = \langle V, E, Ext \rangle$
  - vertices $V$, edges $E \subseteq V^+$
  - external vertices $Ext \subseteq V$

 HR-grammar $G = \langle N, T, R, n^\bullet \rangle$
  - non-/terminals $N / T$
  - initial $n^\bullet \in N$
  - rules $R$

 Rule $R : X \in N \rightarrow \mathcal{H}$
  - $\mathcal{H}$ is hypergraph whose
    (i) vertices are from $N \cup T$
    (ii) $X$ has “arity” $|Ext_\mathcal{H}|$

 Rule-width of $G$: $|\{ \text{vertices of rule's rhs} \}| - 1$
Hypergraph Replacement Grammars

\[ \text{Hypergraph } \mathcal{H} = \langle V, E, \text{Ext} \rangle \]
- vertices \( V \), edges \( E \in V^+ \)
- external vertices \( \text{Ext} \subseteq V \)

\[ \text{HR-grammar } \mathcal{G} = \langle N, T, \mathcal{R}, n^\bullet \rangle \]
- non-/terminals \( N/T \)
- initial \( n^\bullet \in N \)
- rules \( \mathcal{R} \)

\[ \text{rule } R : X \in N \rightarrow \mathcal{H} \]
- \( \mathcal{H} \) is hypergraph whose
  (i) vertices are from \( N \cup T \)
  (ii) \( X \) has “arity” \( |\text{Ext}_\mathcal{H}| \)

\[ \text{rule-width of } \mathcal{G} : \]
\[ |\{\text{vertices of rule’s rhs}\}| - 1 \]
Hyperedge Replacement Grammars

\( \Rightarrow \) Hypergraph \( \mathcal{H} = \langle V, E, \text{Ext} \rangle \)
- vertices \( V \), edges \( E \in V^+ \)
- external vertices \( \text{Ext} \subseteq V \)

\( \Rightarrow \) HR-grammar \( \mathcal{G} = \langle N, T, \mathcal{R}, n^\bullet \rangle \)
- non-/terminals \( N / T \)
- initial \( n^\bullet \in N \)
- rules \( \mathcal{R} \)

\( \Rightarrow \) rule \( R : X \in N \rightarrow \mathcal{H} \)
- \( \mathcal{H} \) is hypergraph whose
  (i) vertices are from \( N \cup T \)
  (ii) \( X \) has “arity” \( |\text{Ext}_\mathcal{H}| \)

\( \Rightarrow \) rule-width of \( \mathcal{G} \):
\[ |\{\text{vertices of rule’s rhs}\}| - 1 \]
Formalizing Anti-/Patterns

[Seese ’91]
If a class of graphs contains all grids (as minors), then we cannot decide MSO over this class.

\[ \text{directly show undecidability} \]
\[ ? \text{ can we generate all } n \times n \text{ grids in } \mathcal{C} \]
example: MSC / CFSM

[Lauteemann ’88]
Every graph generated by a HRG of rule-width \( k \) has treewidth at most \( k \).

\[ \text{easily finding positive results} \]
\[ ? \text{ generate run graph by HRG} \]
\[ ? \text{ additional constraints } \varphi_{rg} \text{ in MSO} \]

Well, let’s start with the real results now...
Formalizing Anti-/Patterns

- [Seese '91]
  If a class of graphs contains all grids (as minors), then we cannot decide MSO over this class.
  ➡️ directly show undecidability
  ➡️ can we generate all $n \times n$ grids in $C$
  example: MSC / CFSM

- [Lautemann '88]
  Every graph generated by a HRG of rule-width $k$ has treewidth at most $k$.
  ➡️ easily finding positive results
  ➡️ generate run graph by HRG
  ➡️ additional constraints $\varphi_{rg}$ in MSO

Well, let’s start with the real results now...
Formalizing Anti-/Patterns

[Seese ’91]

If a class of graphs contains all grids (as minors), then we cannot decide MSO over this class.

⇒ directly show undecidability

? can we generate all $n \times n$ grids in $C$

example: MSC / CFSM

[Lautemann ’88]

Every graph generated by a HRG of rule-width $k$ has treewidth at most $k$.

⇒ easily finding positive results

? generate run graph by HRG

? additional constraints $\varphi_{rg}$ in MSO

Well, let’s start with the real results now...
Formalizing Anti-/Patterns

[Seese '91]
If a class of graphs contains all grids (as minors), then we cannot decide MSO over this class.

- directly show undecidability
  - can we generate all $n \times n$ grids in $C$
example: MSC / CFSM

[Lauteemann '88]
Every graph generated by a HRG of rule-width $k$ has treewidth at most $k$.

- easily finding positive results
  - generate run graph by HRG
  - additional constraints $\varphi_{rg}$ in MSO

Well, let’s start with the real results now…
A HRG for bounded MSC

\[ n^* = 2/2 \quad \mathcal{R} : \quad 2/2 \rightarrow 1/2 \quad \rightarrow \quad 2/1 \quad \rightarrow \quad 0/2 \quad \rightarrow \quad 2/2 \quad \rightarrow \quad 1/1 \]

- **non-terminals** encode remaining queue capacity
- **rhs**: what can happen next
A HRG for bounded MSC

\[ n^* = \begin{array}{c} 2/2 \\ \text{non-terminals} \\ \text{encode remaining} \\ \text{queue capacity} \end{array} \quad \mathcal{R} : \quad \begin{array}{c} 2/2 \\ \end{array} \quad \rightarrow \quad \begin{array}{c} 1/2 \\ \end{array} \quad \rightarrow \quad \begin{array}{c} 0/2 \\ \end{array} \quad \rightarrow \quad \begin{array}{c} 2/2 \\ \rightarrow \end{array} \quad \rightarrow \quad \begin{array}{c} 1/1 \\ \end{array} \quad \rightarrow \quad \begin{array}{c} 1/2 \\ \text{rhs: what can happen next} \end{array} \]
A HRG for bounded MSC

\[ n^* = \begin{array}{c}
2/2 \\
\end{array} \quad \mathcal{R} : \quad \begin{array}{c}
2/2 \\
\end{array} \rightarrow \begin{array}{c}
1/2 \\
0/2 \\
2/2 \\
1/1 \\
\end{array} \]

non-terminals encode remaining queue capacity

rhs: what can happen next
A HRG for bounded MSC

\[ n^* = \begin{array}{c}
\text{non-terminals}
\end{array} \quad \begin{array}{c}
\text{encode remaining}
\end{array} \quad \begin{array}{c}
\text{queue capacity}
\end{array} \]

\[ R : \begin{array}{c}
2/2
\end{array} \rightarrow \begin{array}{c}
1/2
\end{array} \rightarrow \begin{array}{c}
0/2
\end{array} \rightarrow \begin{array}{c}
2/2
\end{array} \rightarrow \begin{array}{c}
1/1
\end{array} \]

rhs: what can happen next
A HRG for bounded MSC

\[ n^* = \begin{array}{c} 2/2 \end{array} \quad \mathcal{R} : \begin{array}{c} 2/2 \end{array} \rightarrow \begin{array}{c} 1/2 \end{array} \quad \begin{array}{c} 2/1 \end{array} \rightarrow \begin{array}{c} 0/2 \end{array} \quad 2/2 \rightarrow 1/1 \]

non-terminals encode remaining queue capacity

rhs: what can happen next
A HRG for bounded MSC

\[ n^* = \begin{array}{c}
\text{non-terminals encode remaining queue capacity}
\end{array} \]

\[ \mathcal{R} : \begin{array}{c}
\text{rhs: what can happen next}
\end{array} \]

\[ \begin{array}{c}
2/2 \\
2/2 \\
\vdots \\
1/2 \\
1/2 \\
\vdots \\
0/2 \\
0/2 \\
\vdots \\
2/1 \\
2/1 \\
\vdots \\
2/2 \\
2/2 \\
\vdots \\
1/1 \\
1/1
\end{array} \]
A HRG for bounded MSC

$$n^* = \frac{2}{2}$$  \(\mathcal{R}: \frac{2}{2} \rightarrow \frac{1}{2} \frac{2/1}{0/2} \frac{2/2}{1/1} \ldots\)

- non-terminals encode remaining queue capacity

rhs: what can happen next
A HRG for bounded MSC

\[ n^* = \begin{array}{c}
\begin{array}{c}
\text{2/2}
\end{array}
\end{array} \quad \mathcal{R} : \quad \begin{array}{c}
\begin{array}{c}
\text{2/2}
\end{array}
\end{array} \rightarrow \begin{array}{c}
\begin{array}{c}
\text{1/2}
\end{array}
\end{array} \rightarrow \begin{array}{c}
\begin{array}{c}
\text{0/2}
\end{array}
\end{array} \rightarrow \begin{array}{c}
\begin{array}{c}
\text{2/2}
\end{array}
\end{array} \rightarrow \begin{array}{c}
\begin{array}{c}
\text{1/1}
\end{array}
\end{array}
\end{array} \]

- HRG generates run graphs of 2-bounded MSC with 2 processes
- generalize: \( p \) processes and bound \( k \)
- treewidth of run graphs of bounded MSC is \( \mathcal{O}(p \cdot k) \)
Lock (Causality) Graphs

- Induced Edges: These induced edges are key in guaranteeing soundness and deduce the new causality constraint constraints that are captured as location it is not released until after.

- Synchronization Seed Edges: Synchronization

- Lock (Causality) Graphs

- Fork-Join
- Wait-Notify

- An Example Universal Causality Graph

Fig. 1. An Example Universal Causality Graph

- introduced in [Kahlon/Wang ’10]
- encode in non-terminals who holds and who requests lock
- add additional back-causality edges (from last holder to new)
- possible in $O(p \cdot l)$ for $p$ processes and $l$ locks
decidability depends on the following restrictions [HLMS ’10]
take a closer look at interplay of pushdowns and sync
take a closer look at interplay of pushdowns and sync
take a closer look at interplay of pushdowns and sync
⇒ take a closer look at interplay of pushdowns and sync
“cut” the run to simulate it on one pushdown

more general class of 1-pd-simulateable systems in [Atig ’10]
a HRG for the Two Process Setting

\[ n^* = \begin{array}{c}
\text{X}
\end{array} \]

\[ \mathcal{R} : \]

- \[ \text{X} \rightarrow p(q) \text{X} \]
- \[ p(q) \rightarrow p(q) p(q) \]
- \[ p(q) \rightarrow \begin{array}{c}
p(q)
p(d)
\end{array} \]
- \[ \text{pd} \rightarrow \text{pd} \]

use of pd restricted for process q, unrestricted for proc. p
...a HRG for the Two Process Setting

\( n^* = \bullet \quad X \quad \text{use of pd restricted for process } q, \text{ unrestricted for proc. } p \)

\[ \mathcal{R} : \]

\[ \bullet \quad X \quad \leftrightarrow \quad p(q) \quad X \quad \leftrightarrow \quad (p)q \quad X \]

\[ p(q) \quad \leftrightarrow \quad p(q) \quad p(q) \quad \text{generalize to } p \text{ procs.: rule-width remains bounded by } O(p) \]

\[ \bullet \quad pd \quad p(q) \quad \leftrightarrow \quad pd \quad pd \quad \leftrightarrow \quad pd \]
Summary/Outlook

- retrace known results in a simple “visual” way (*intuitions!*)
- give several new results of MSO-decidable QCP classes
- framework to “pre-test” restrictions for QCP wrt. decidability

- more detailed complexity results missing in approach
- what about “beneath the stars”: \(\mu\)-calculus, LTL, simple reachability, and their relation to graph grammars?
- from MSO on *one* run to *branching* runs?
- can we extend these ideas to a dynamic setting?
- catalogue of anti-patterns?

- please feel free to append your ideasremarks to this list...