Verification of Run Graphs

(Some Preliminary Results)

Alexander Heußner

LaBRI Bordeaux, France

Distributed Systems

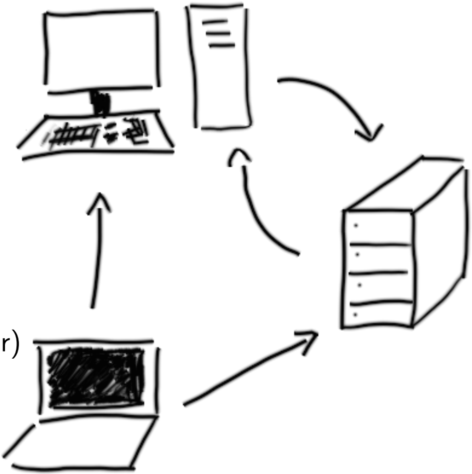
"A distributed system consists of a collection of distinct processes which are spatially separated & communicate with one another by exchanging messages."

- 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

Distributed Systems

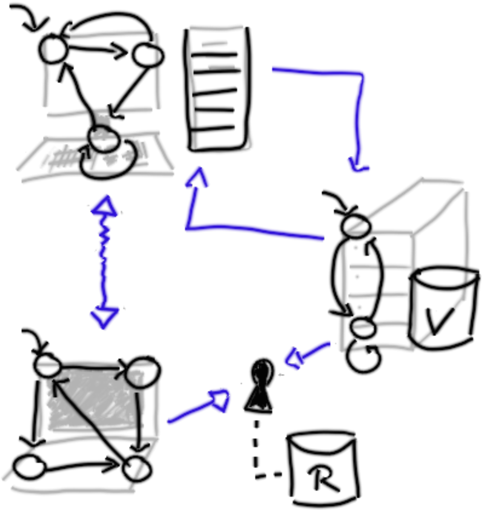
"A distributed system consists of a collection of distinct processes which are spatially separated & communicate with one another by exchanging messages."

- ➪ 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



(Q)ueueing (C)ommunicating (P)rocesses

- 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
  - ...



Recap: Ongoing Research

- fifo queues (I): Message Sequence Charts MSC
  - bounded branching of unfolding [Madhusudan '01]
  - ∃/∀ bounded channels [Lohrey/Musholl '04]
  - ...

semantic restrictions!

- 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]
  - ∃/∀ 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!

add push-downs

Recap: Ongoing Research

- fifo queues (I): Message Sequence Charts MSC
  - bounded branching of unfolding [Madhusudan '01]
  - ∃/∀ bounded channels [Lohrey/Musholl '04]
  - . . .

semantic restrictions!

add push-downs

- 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

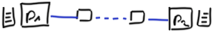
- fifo queues (I): Message Sequence Charts MSC
  - bounded branching of unfolding [Madhusudan '01]
  - ∃/∀ bounded channels [Lohrey/Musholl '04]
  - ...

semantic restrictions!

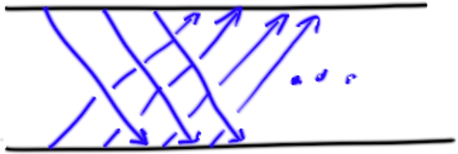
add push-downs

- 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]

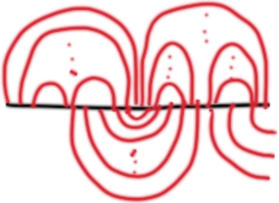
- fifo queues (III) [Heußner/Leroux/Muscholl/Sutre '10]
  - generalize previous stack-queue-interplay restriction
  - focus atomic send-receives (semantic restriction)
  - communication architecture does not allow 

Antipatterns/“Contra”-Patterns

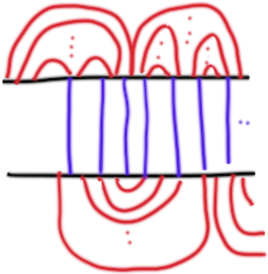


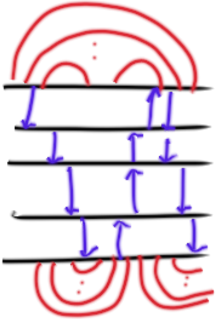
[Brand/Zafiropoulo ’83]

- distinctive causal entanglement patterns lead to undecidability issues

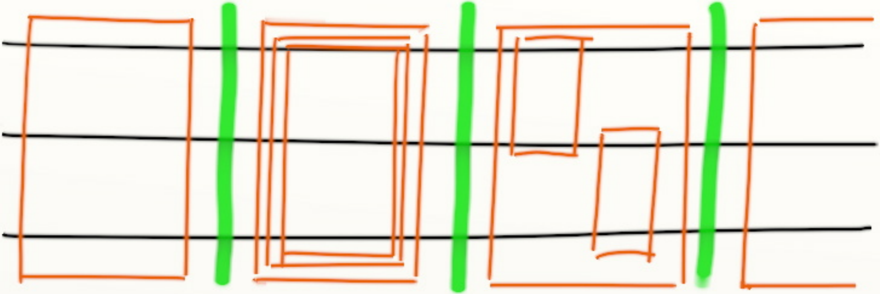


Multi-stack PDA





Decidability/"Pro"-Patterns



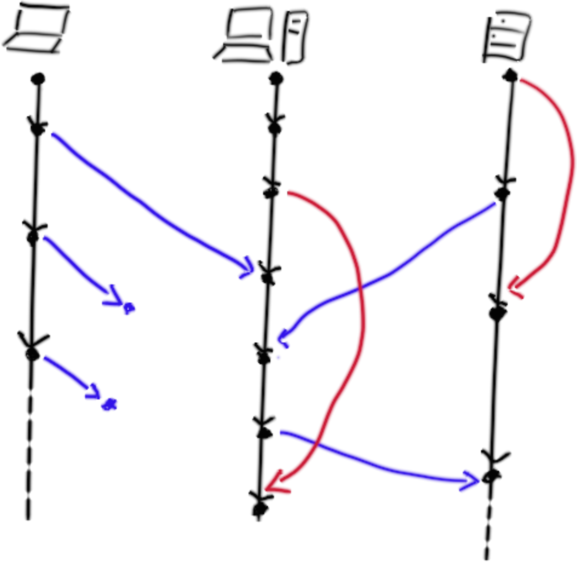
- cut run into sequence of sub-patterns
- only "exchange" finite information between these patterns
- sub-patterns either use recursion or composition of sub-sub-patterns
- "context-freeness"

How to Bridge The Gap ?

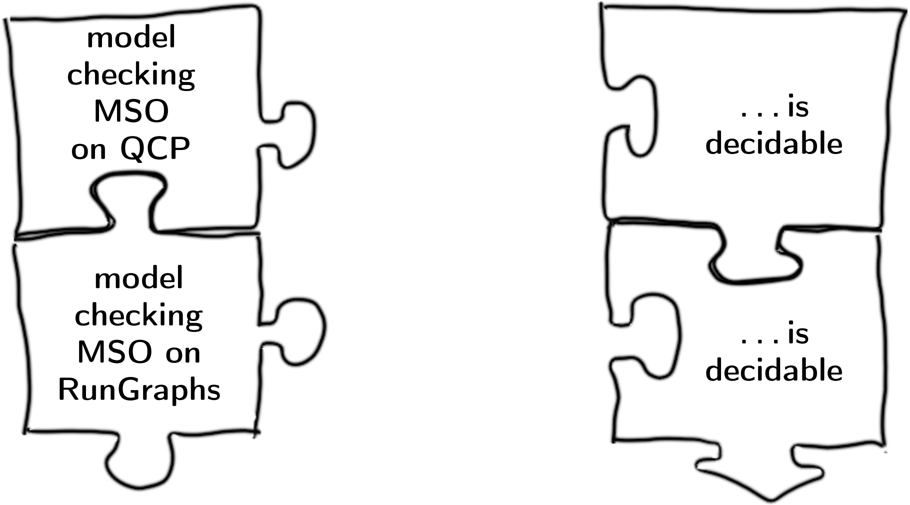
Run Graphs[1]

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

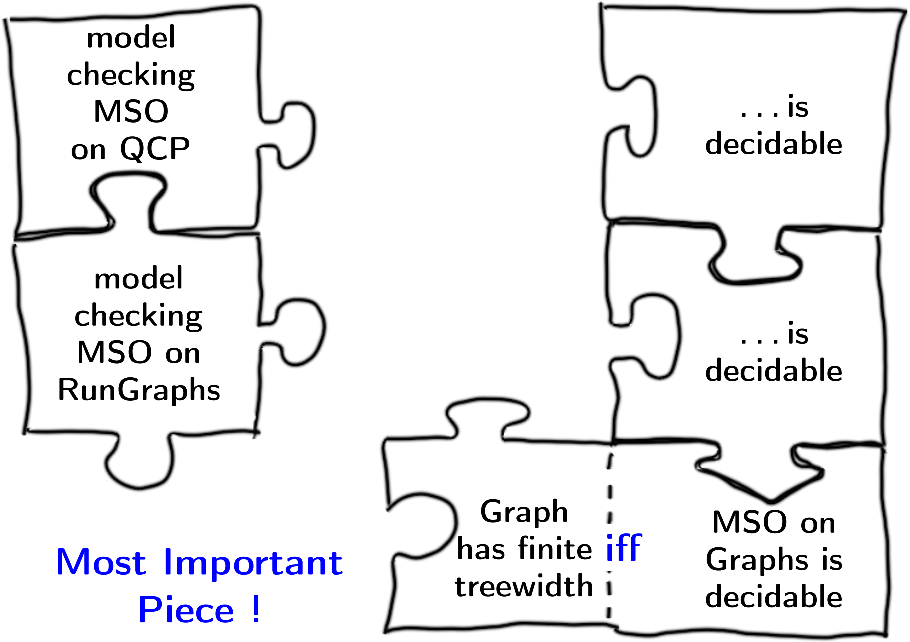
hence,
partial order semantics !



[1] aka "space-time diagrams", extended Hasse diagrams, MSC with pushdowns,. . .



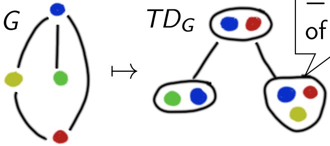
“Easier” & more Concise Question, but still. . .


model checking MSO on QCP
model checking MSO on RunGraphs
Most Important Piece !
. . . is decidable
. . . is decidable
Graph has finite treewidth
iff
MSO on Graphs 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

treewidth $=$ max size of "bag" -1



x

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

[Courcelle's Theorem]

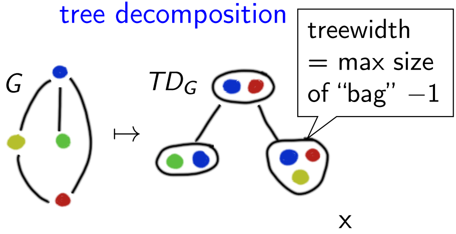
SAT(MSO) is fixed parameter tractable for given treewidth of model and size of formula.

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

Given class $\mathcal{C}$ of graphs of bounded treewidth which are MSO definable, and an MSO formula φ, then we can test whether there exists $G \in \mathcal{C}$ 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



- ➪ 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.

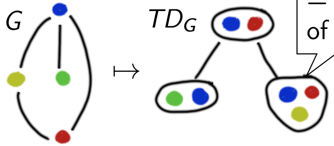
[Madhusudan/Parlato '10(?)] based on [Seese '91]

Given class $\mathcal{C}$ of graphs of bounded treewidth which are MSO definable, and an MSO formula φ, then we can test whether there exists $G \in \mathcal{C}$ 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

treewidth = max size of "bag" -1



x

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

[Courcelle's Theorem]

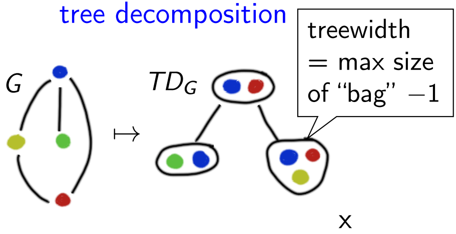
SAT(MSO) is fixed parameter tractable for given treewidth of model and size of formula.

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

Given class $\mathcal{C}$ of graphs of bounded treewidth which are MSO definable, and an MSO formula φ, then we can test whether there exists $G \in \mathcal{C}$ 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



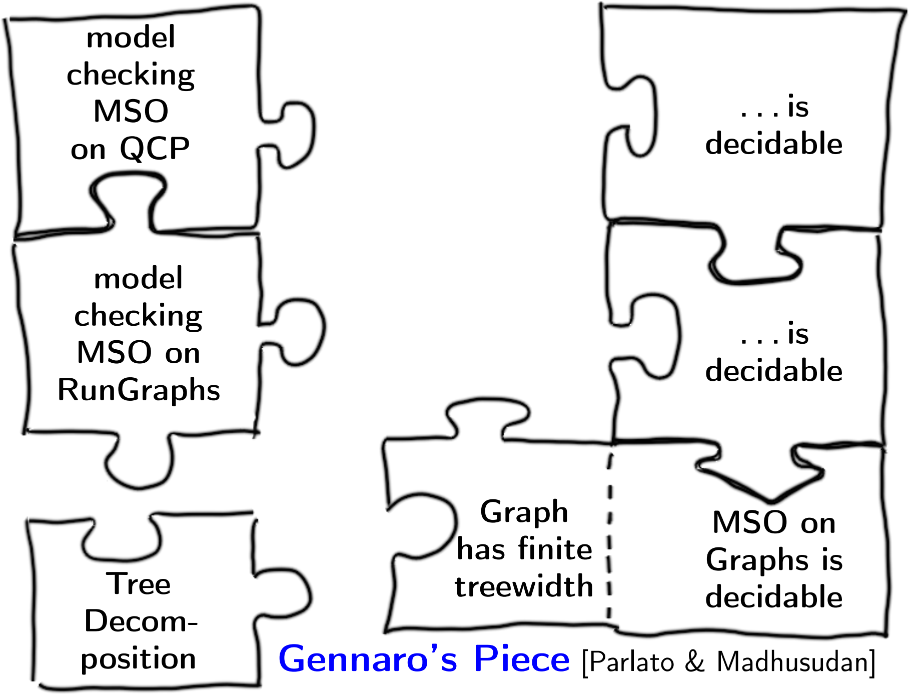
- apply (some) tree-based algos to general graphs via TD_G
- extension to classes of graphs leads to unbounded treewidth

[Courcelle's Theorem]

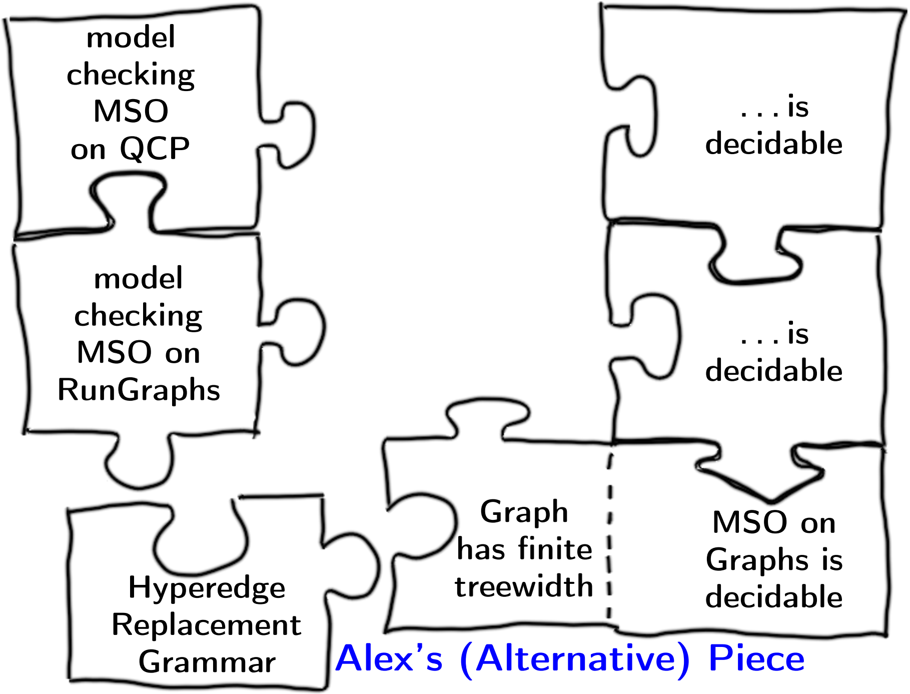
SAT(MSO) is fixed parameter tractable for given treewidth of model and size of formula.

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

Given class $\mathcal{C}$ of graphs of bounded treewidth which are MSO definable, and an MSO formula φ, then we can test whether there exists $G \in \mathcal{C}$ s.t. $G \models \varphi$.

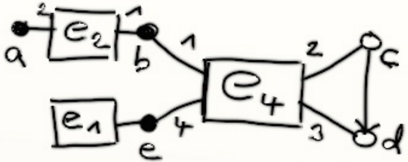


Gennaro's Piece [Parlato & Madhusudan]



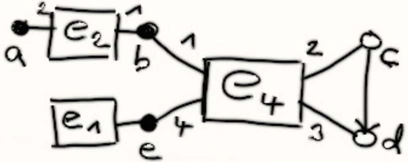
Alex's (Alternative) Piece

Hyperedge Replacement Grammars



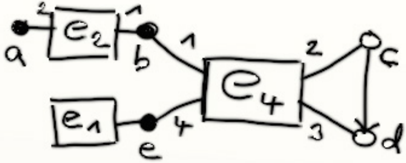
- *Hypergraph* $\mathcal{H} = \langle V, E, Ext \rangle$
  - vertices V, edges $E \in V^+$
  - external vertices $Ext \subseteq V$
- HR-grammar $\mathcal{G} = \langle N, T, \mathcal{R}, n^\bullet \rangle$
  - non-/terminals N/T
  - initial $n^\bullet \in N$
  - rules $\mathcal{R}$
- rule $R : X \in N \hookrightarrow \mathcal{H}$
  - $\mathcal{H}$ is hypergraph whose
    - (i) vertices are from $N \cup T$
    - (ii) X has "arity" $|Ext_{\mathcal{H}}|$
- rule-width of $\mathcal{G}$:
  $|\{\text{vertices of rule's rhs}\}| - 1$

Hyperedge Replacement Grammars



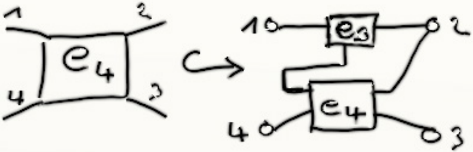
- *Hypergraph* $\mathcal{H} = \langle V, E, Ext \rangle$
  - vertices V, edges $E \in V^+$
  - external vertices $Ext \subseteq V$
- HR-grammar $\mathcal{G} = \langle N, T, \mathcal{R}, n^\bullet \rangle$
  - non-/terminals N/T
  - initial $n^\bullet \in N$
  - rules $\mathcal{R}$
- rule $R : X \in N \hookrightarrow \mathcal{H}$
  - $\mathcal{H}$ is hypergraph whose
    - (i) vertices are from $N \cup T$
    - (ii) X has "arity" $|Ext_{\mathcal{H}}|$
- rule-width of $\mathcal{G}$:
  $|\{\text{vertices of rule's rhs}\}| - 1$

Hyperedge Replacement Grammars



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

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

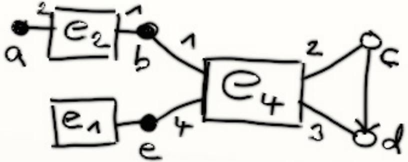


- rule $R : X \in N \hookrightarrow \mathcal{H}$
  - $\mathcal{H}$ is hypergraph whose
    - (i) vertices are from $N \cup T$
    - (ii) X has "arity" $|Ext_{\mathcal{H}}|$

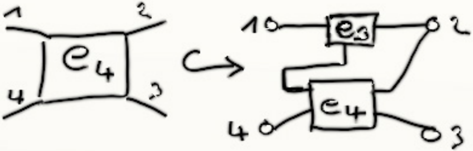
- rule-width of $\mathcal{G}$:
  $|\{\text{vertices of rule's rhs}\}| - 1$

Hyperedge Replacement Grammars

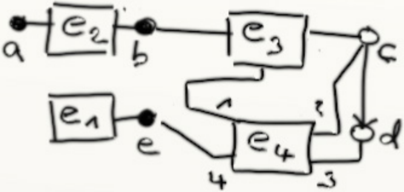
- *Hypergraph* $\mathcal{H} = \langle V, E, Ext \rangle$
  - vertices V, edges $E \in V^+$
  - external vertices $Ext \subseteq V$



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

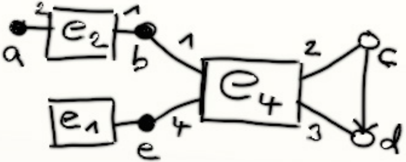


- rule $R : X \in N \hookrightarrow \mathcal{H}$
  - $\mathcal{H}$ is hypergraph whose
    - (i) vertices are from $N \cup T$
    - (ii) X has "arity" $|Ext_{\mathcal{H}}|$

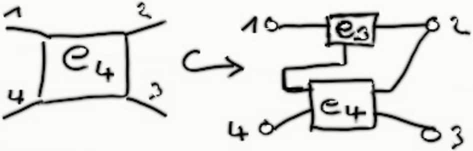


- rule-width of $\mathcal{G}$: $|\{\text{vertices of rule's rhs}\}| - 1$

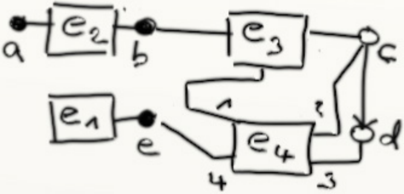
Hyperedge Replacement Grammars



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



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



- ➪ rule $R : X \in N \hookrightarrow \mathcal{H}$
  - $\mathcal{H}$ is hypergraph whose
    - (i) vertices are from $N \cup T$
    - (ii) X has "arity" $|Ext_{\mathcal{H}}|$
- ➪ 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.

[Lautemann '88]

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

➪ directly show undecidability

- ? can we generate all $n \times n$ grids in $\mathcal{C}$

example: MSC / CFSM

➪ easily finding positive results

- ? generate run graph by HRG
- ? additional constraints φ_{rg} in MSO

Well, let's start with the real results now. . .

Formalizing Anti-/Patterns

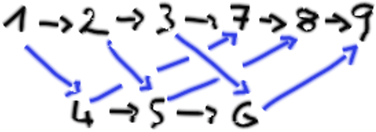
[Seese '91]

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 $\mathcal{C}$

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 φ_{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.

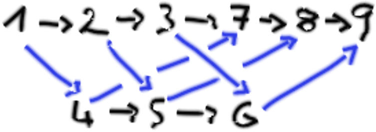
[Lautemann '88]

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

➪ directly show undecidability

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

example: MSC / CFSM



➪ easily finding positive results

? generate run graph by HRG

? additional constraints φ_{rg} in MSO

Well, let's start with the real results now. . .

Formalizing Anti-/Patterns

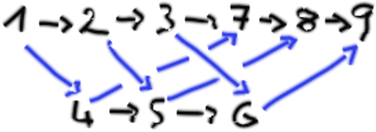
[Seese '91]

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 $\mathcal{C}$

example: MSC / CFSM



[Lautemann '88]

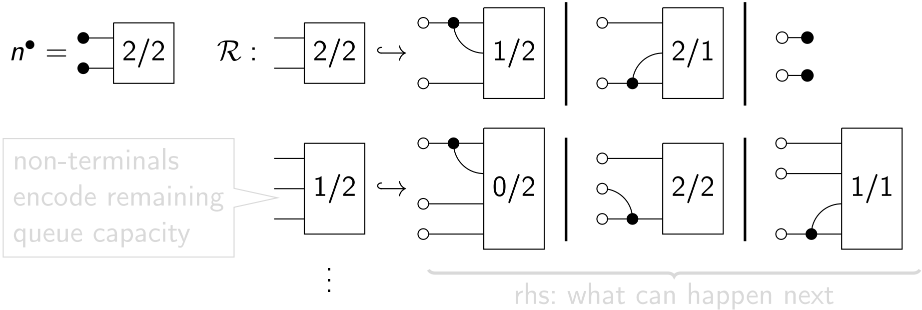
Every graph generated by a HRG of rule-width k has treewidth at most k.

➪ easily finding positive results

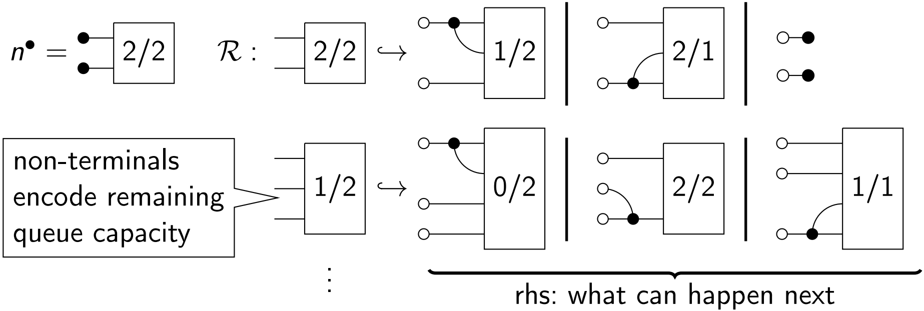
- ? generate run graph by HRG
- ? additional constraints φ_{rg} in MSO

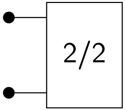
Well, let's start with the real results now. . .

A HRG for bounded MSC

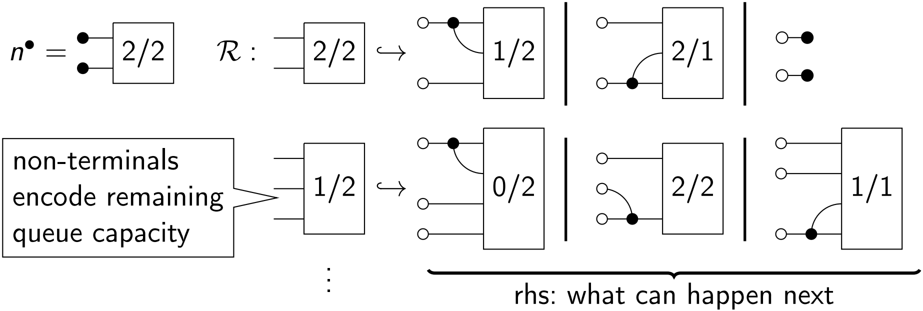


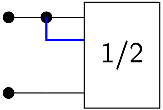
A HRG for bounded MSC



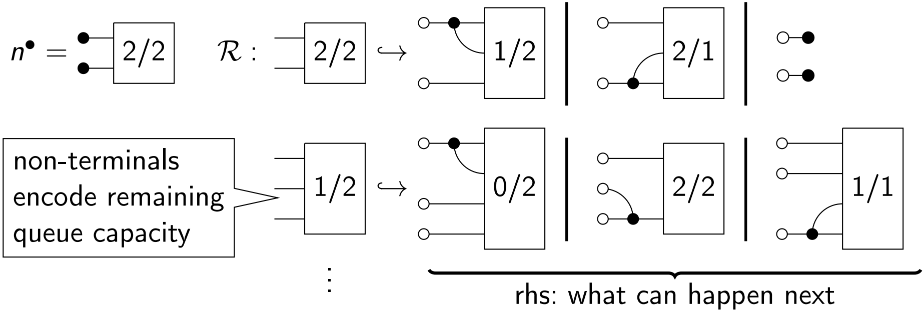


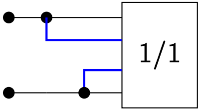
A HRG for bounded MSC



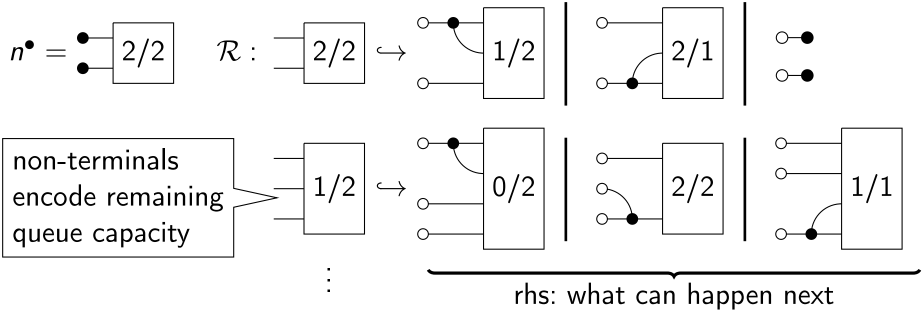


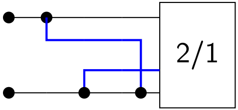
A HRG for bounded MSC



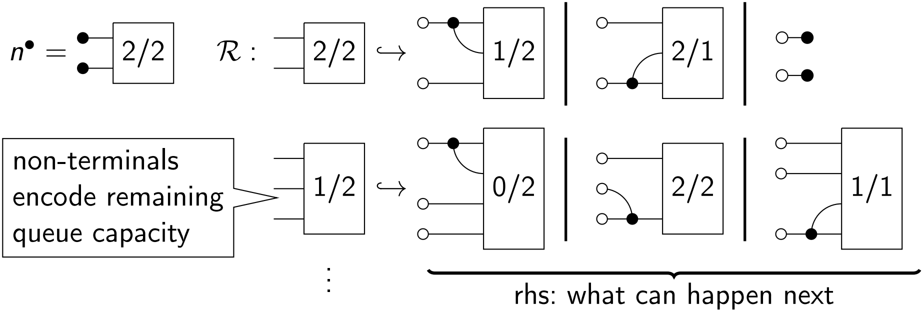


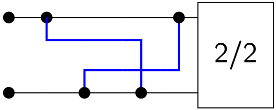
A HRG for bounded MSC



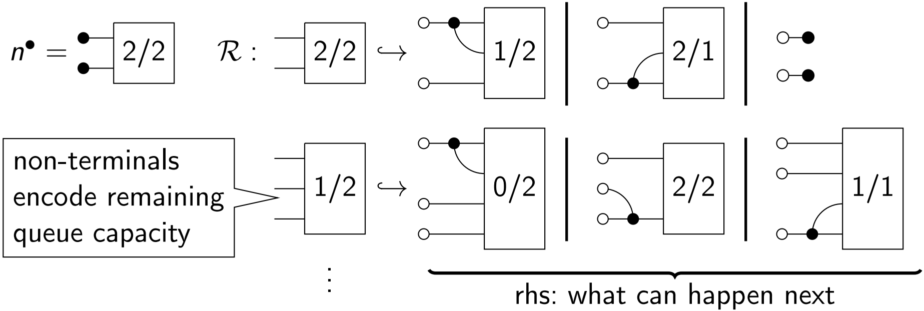


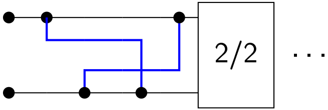
A HRG for bounded MSC



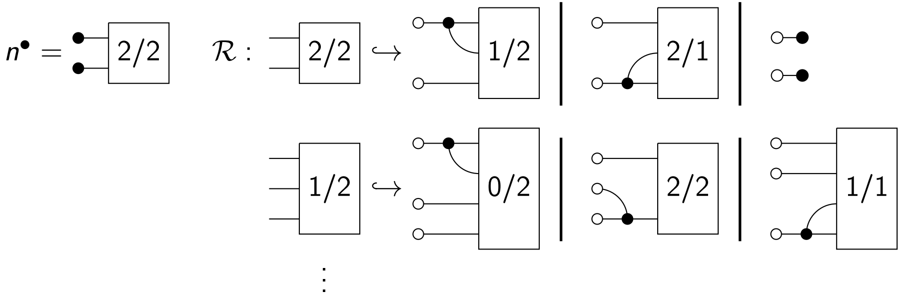


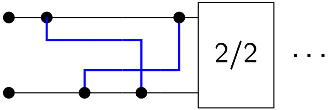
A HRG for bounded MSC





A HRG for bounded MSC





- 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

T_1	T_2
a0: lock(l_3);	b0: lock(l_1);
a1: lock(l_1);	b1: notify(c);
a2: wait$_{pre}$(c)	b2: unlock(l_1);
a3: unlock(l_1)	b3: lock(l_1);
a4: lock(l_1)	b4: lock(l_3);
a5: wait$_{post}$(c);	b5: unlock(l_1);
a6: lock(l_2);	b6: lock(l_2);
a7: unlock(l_3);	b7: unlock(l_2);
a8: unlock(l_1);	b8: unlock(l_3);
a9: $sh = sh + 1$;	b9: $sh = sh + 2$;
a10: unlock(l_2);	b10: ...

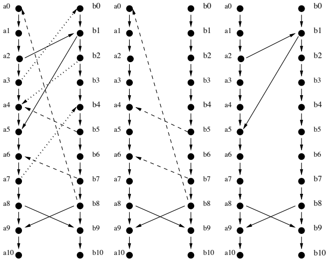


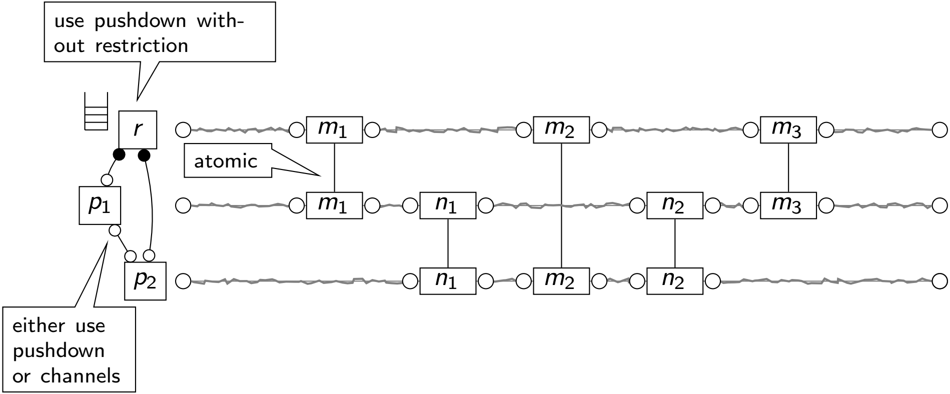
Fig. 1. An Example Universal Causality Graph

figure taken from their CAV'10 paper

- 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 $\mathcal{O}(p \cdot l)$ for p processes and l locks

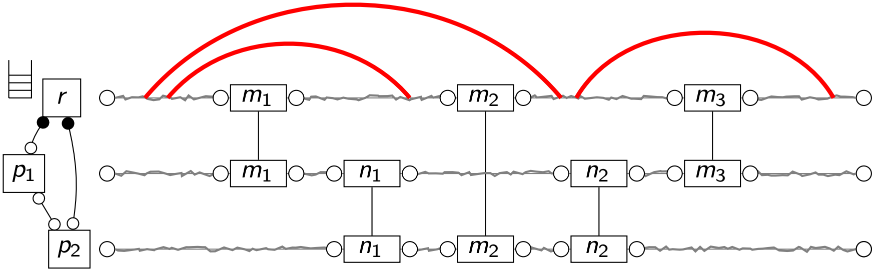
Well-queueing Eager Non-confluent RQCP

➪ decidability depends on the following restrictions [HLMS '10]



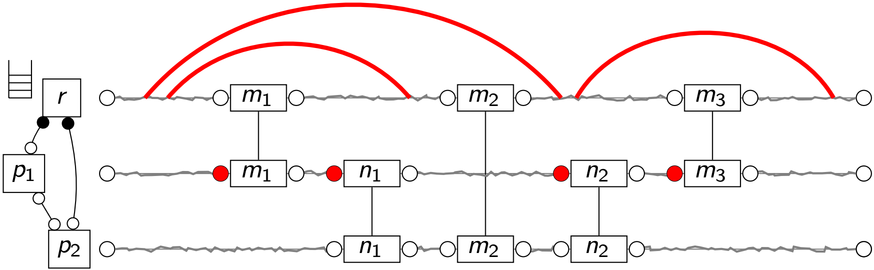
Well-queueing Eager Non-confluent RQCP

➪ take a closer look at interplay of pushdowns and sync



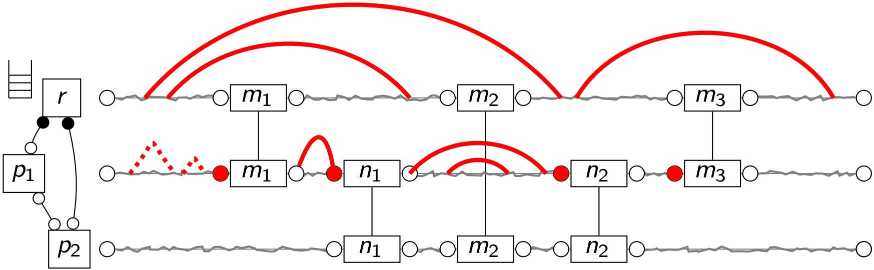
Well-queueing Eager Non-confluent RQCP

➪ take a closer look at interplay of pushdowns and sync



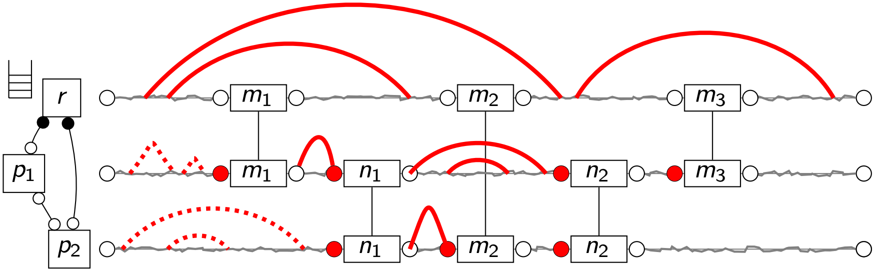
Well-queueing Eager Non-confluent RQCP

➪ take a closer look at interplay of pushdowns and sync



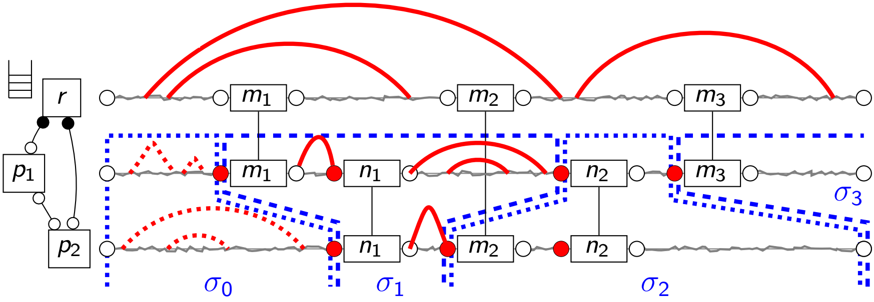
Well-queueing Eager Non-confluent RQCP

➪ take a closer look at interplay of pushdowns and sync



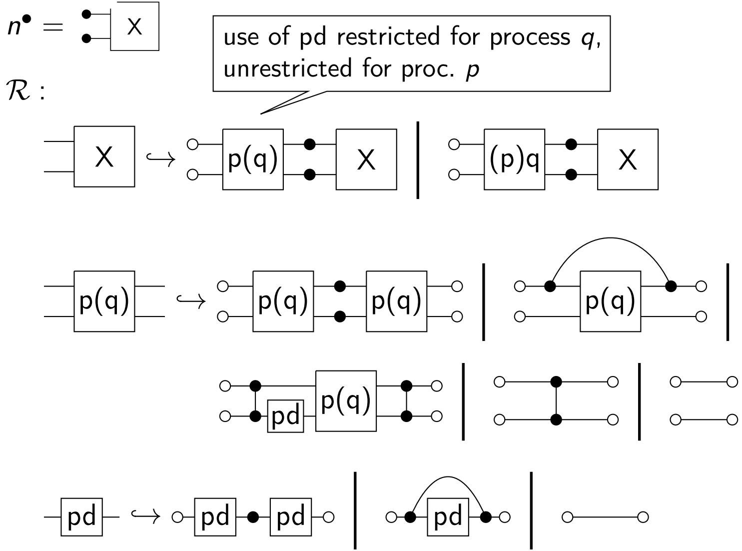
Well-queueing Eager Non-confluent RQCP

➪ "cut" the run to simulate it on one pushdown

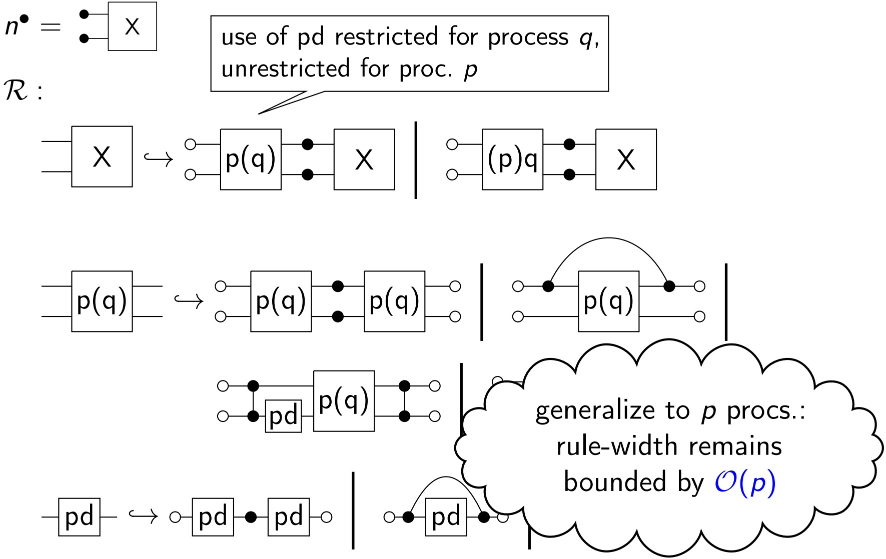


➪ more general class of 1-pd-simulateable systems in [Atig '10]

. . . a HRG for the Two Process Setting



. . . a HRG for the Two Process Setting



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": μ-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 ideas/remarks to this list. . .