The Tree Width of Auxiliary Storage

Gennaro Parlato (LIAFA, CNRS, Paris, France)

joint work with

P. Madhusudan (Univ of Illinois at Urbana-Champaign, USA)

Automata with aux storage

- Turing machines = finite automata + 1 infinite tape
  undecidable emptiness

- CFLs = finite automata + 1 stack
  Decidable emptiness

- Finite automata + 2 stacks; Finite automata + 1 queue
  → undecidable emptiness

Studies into finding the “boundary” of decidability

Verification:
renewed interest in automata+aux storage

- Static analysis and dataflow analysis of control flow
  - Context-sensitive static data-flow analysis of programs with recursion is essentially PDA emptiness
  - Reps-Horowitz-Sagiv; Sharir-Pnueli

- SLAM project from Microsoft Research
  - Predicate abstraction of software that abstracts data to Boolean domains [Ball-Rajamani-'90]
  - Constructs a PDA model and checks emptiness/reachability.
  - **BEBOP:** PDA emptiness using "summaries"
    MOPED: PDA emptiness using regularity of reachable configs
    GETAFIX: PDA emptiness using fixed-points

- Concurrent program verification using abstraction
  - Emptiness of multi-stack automata
  - Emptiness of distributed automata with FIFO queues

Decidable emptiness

Multistack pushdown automata with

- k-context-switches (Qadeer , Rehof- TACAS'05)
- k-phases (La Torre, Madhusudan, Parlato - LICS'07)
- ordered (Breveglieri, Cherubini, Citrini, Crespi-Reghizzi – JFOCS'95)
- Parameterized pushdown automata with k contexts (La Torre, Madhusudan, Parlato - CAV'10)

Distributed automata with FIFO queues

- finite state processes with polyforest architecture
- pushdown processes with forest architecture + well queuing (La Torre, Madhusudan, Parlato - TACAS'08)
- Non-confluent architectures + size 1 queues (Heußner, Leroux, Muscholl, Sutre - FOSSACS'10)

Decidable emptiness

Multistack pushdown automata with

- k-context-switches
- k-phases
- ordered (P
- Para

TACAS'05)
- LICS'07)
JFOCS'95)
CAV'10)

Distri

- f
- p

itecture
rest architecture + well queuing
(La Torre, Madhusudan, Parlato - TACAS'08)

- No fluent architectures + size 1 queues
(Heußner, Leroux, Muscholl, Sutre - FOSSACS'10)

Uniform property: Awkward definitions; fragile algorithms

The question is …

So many decidable subclasses; so much awkwardness

Is there a robust common principle that explains their decidability?

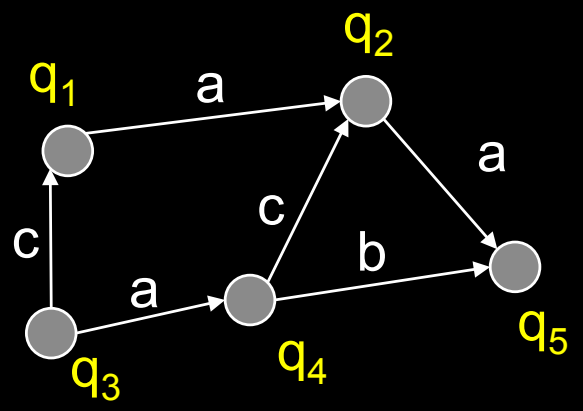
Answer:

We present a general criterion that uniformly explains many of such results:

Decidable ← Simulate by graph automata working on graphs of bounded tree width

Graph automata

- Fix a class *C* of Σ-labeled graphs
- A graph automaton over *C* is a triple GA = (Q, tiles)
  - *Q* is a finite set of states
  - tiles $\subseteq Q \times \Sigma \times Q$



- A graph G is accepted by GA if we can decorate each node with a state such that
  - Each edge can be tiled

Tree-width of graphs G=(V,E)

A tree decomposition of *G* is a pair $(T, \{bag_z\}_{z \text{ is a node of } T})$, where T is a tree and bag_z *is a subset of G nodes, such that*

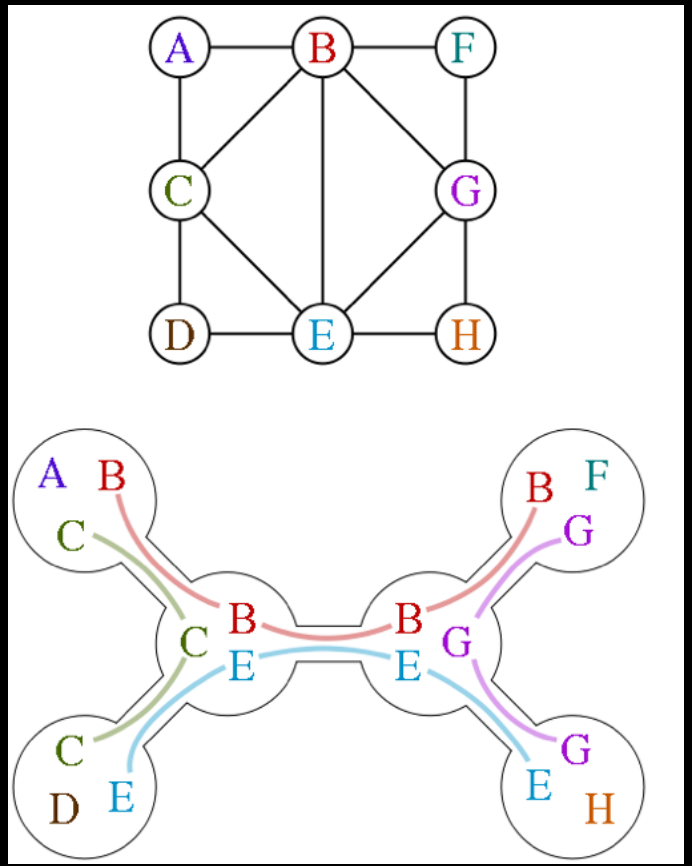
- every edge of *G* has both endpoints in some bag
- for every node of *G* the set of bags that contain it form a subtree of *T*

The **width** of the tree decomposition is the maximum of $|bag_z|$-1 over all T nodes *z*

The **treewidth** of a graph *G*, is the minimum such width over all tree decompositions of *G*

Tree width of a tree is 1 ☺

Tree width of a k-clique is k-1.



Monadic second-order logic on graphs

MSO is given by the following syntax

$$\varphi ::= x=y \mid E_a(x,y) \mid x \in Z \mid \varphi \vee \varphi \mid \neg\varphi \mid \exists x.\, \varphi(x) \mid \exists Z.\, \varphi(Z)$$

where

- *x, y* are first-order variables, *Z* is a second-order variable
- $E_a(x,y)$ is a binary edge relation

- A class of Σ-labeled graphs *C* is MSO-definable if there is a φ such that

$$C = \{ G \mid G \text{ satisfies } \varphi \}$$

MSO on graphs + GA emptiness

- Over the class of all graphs, MSO satisfiability is undecidable
  (even FO is undecidable)

- The satisfiability problem for MSO on the class of all graphs of tree width k, for any k, is decidable. [Courcelle]

- If C is any class of graphs of
  - bounded tree-width &
  - MSO definable,

  then satisfiability of MSO on C is decidable.

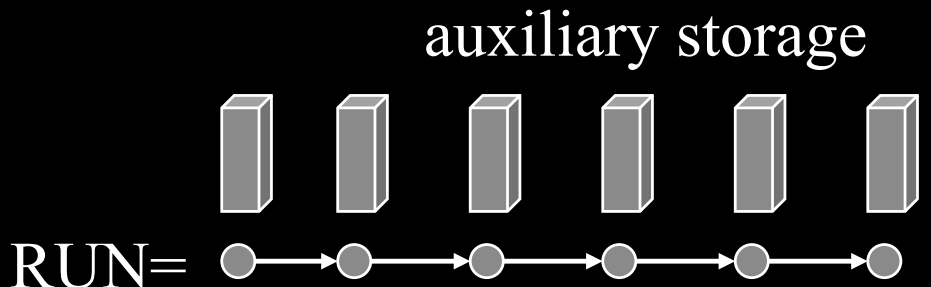
- Graph automata emptiness on C → MSO satisfiability on C
- Hence graph automata emptiness on any class C of MSO definable graphs of bounded tree-width bdd tw is decidable.

Main schema for decidability

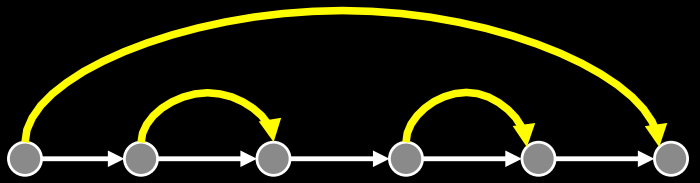
Emptiness
Automaton with
auxiliary storage

Reduction

Emptiness
Graph Automaton over a class C
- MSO definable
-Bounded treewidth



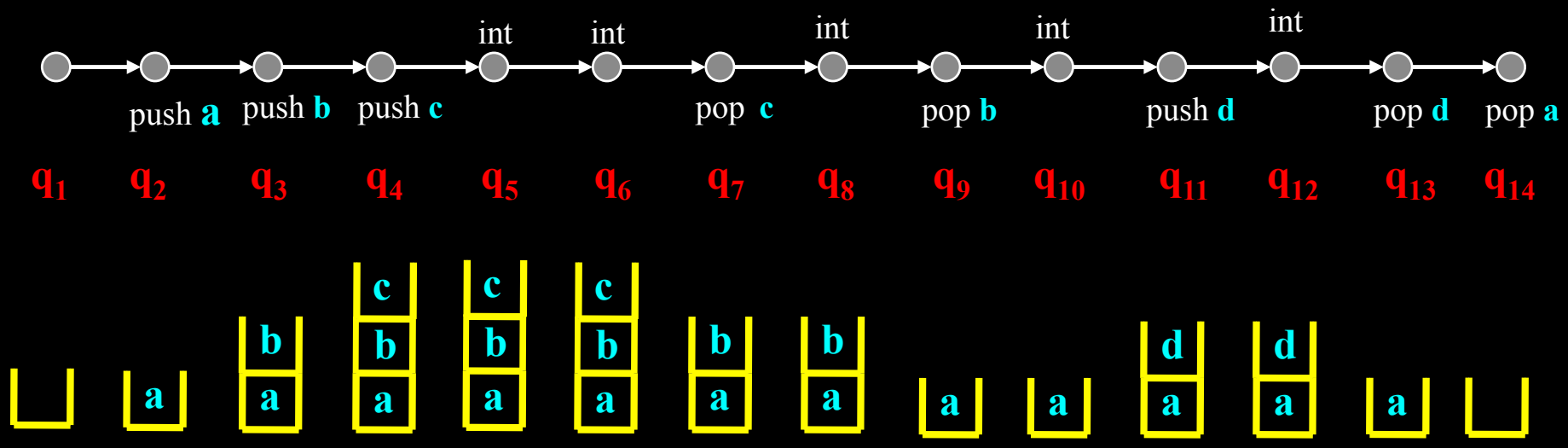
Compile down
the auxiliary storage
in the graph



Simple graph
(no auxiliary storage)

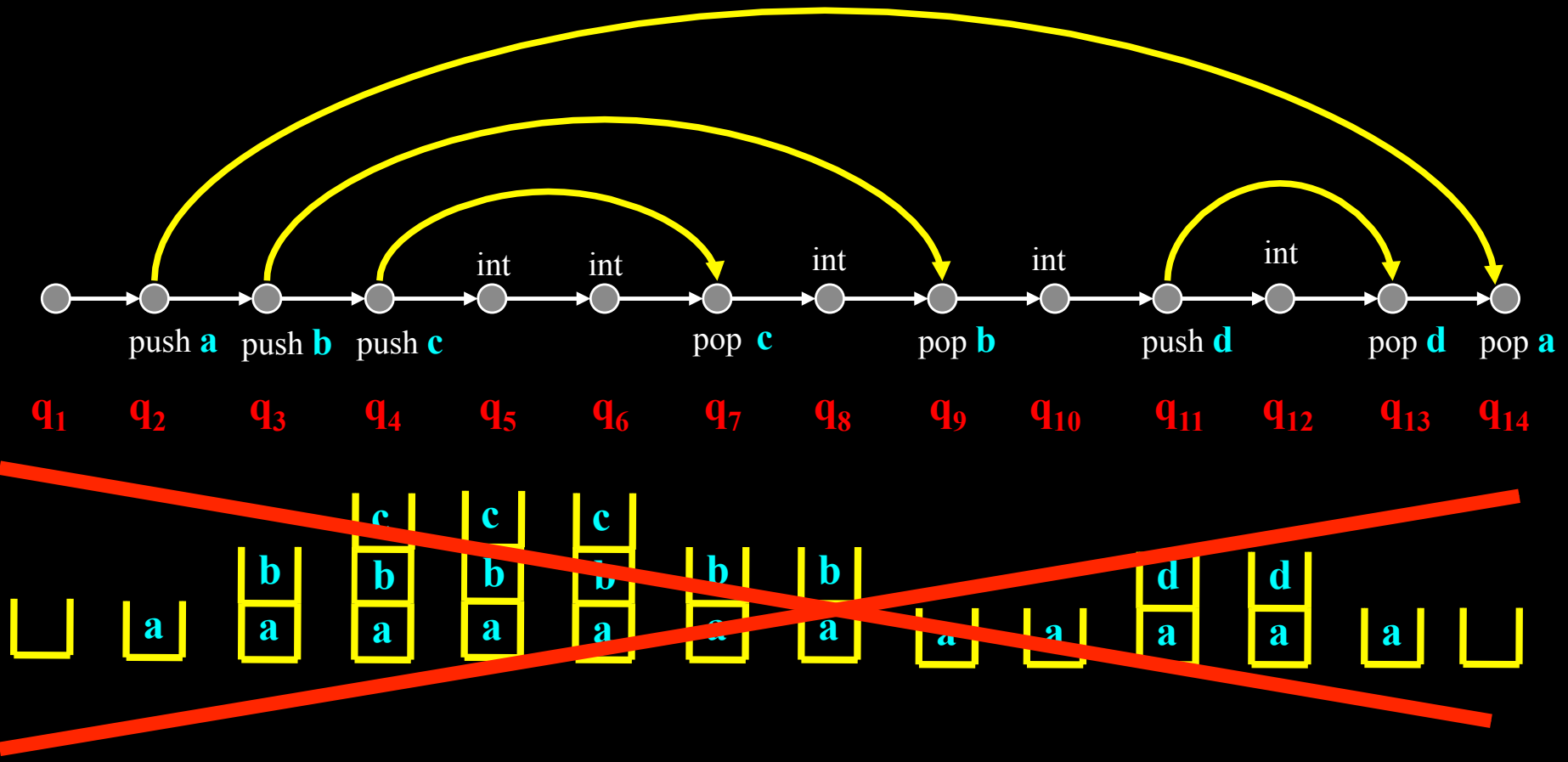
Simulation
for
pushdown automata

PDA ←→ Nested words



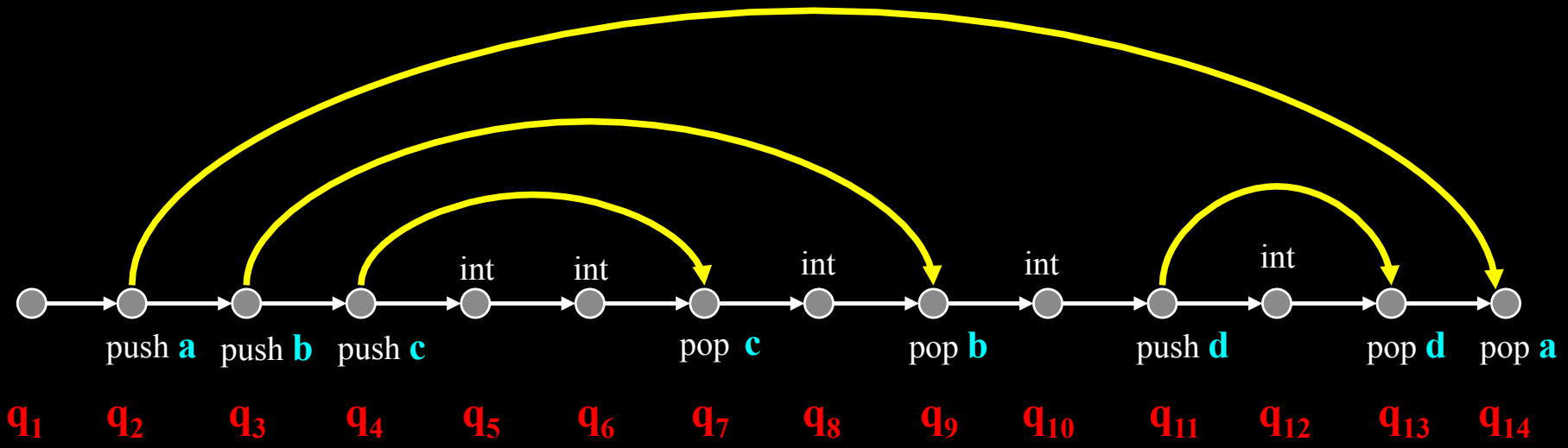
RUN

PDA ←→ Nested words



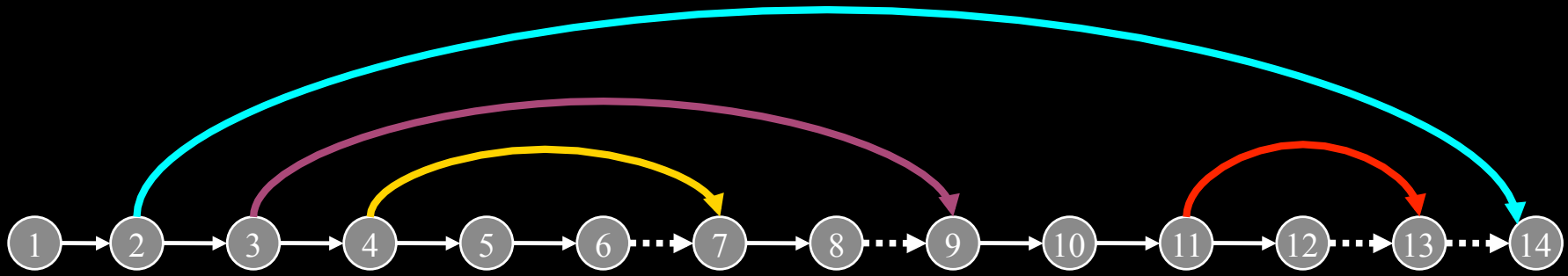
RUN

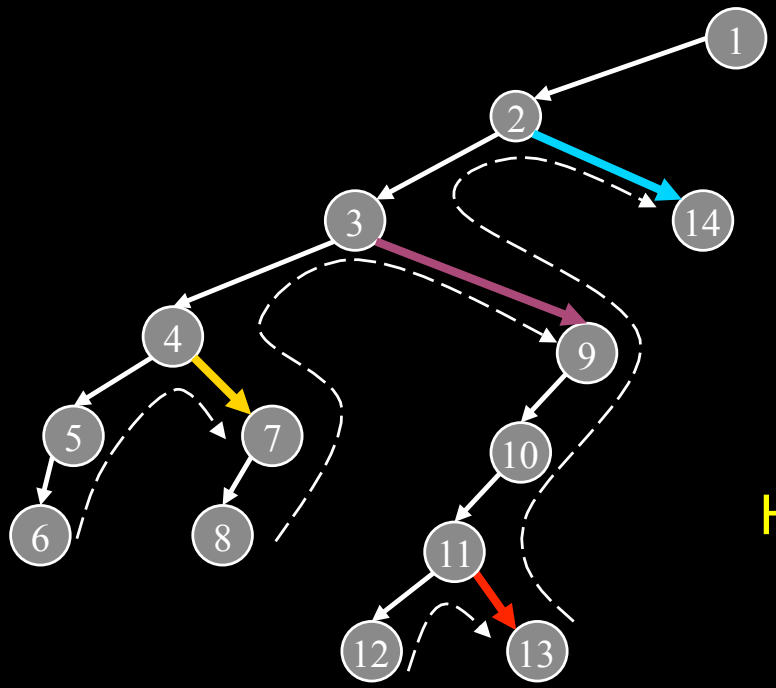
PDA ←→ Nested words



- A NW graph captures the behavior of a run
  - The stack is compiled down into the nested word (nesting edges)
- The class of NWs is MSO definable
  - linear order + nesting edges; nesting edges should not cross
- Graph automata working over this graph can simulate a PDA

Nested words have tree-width 2

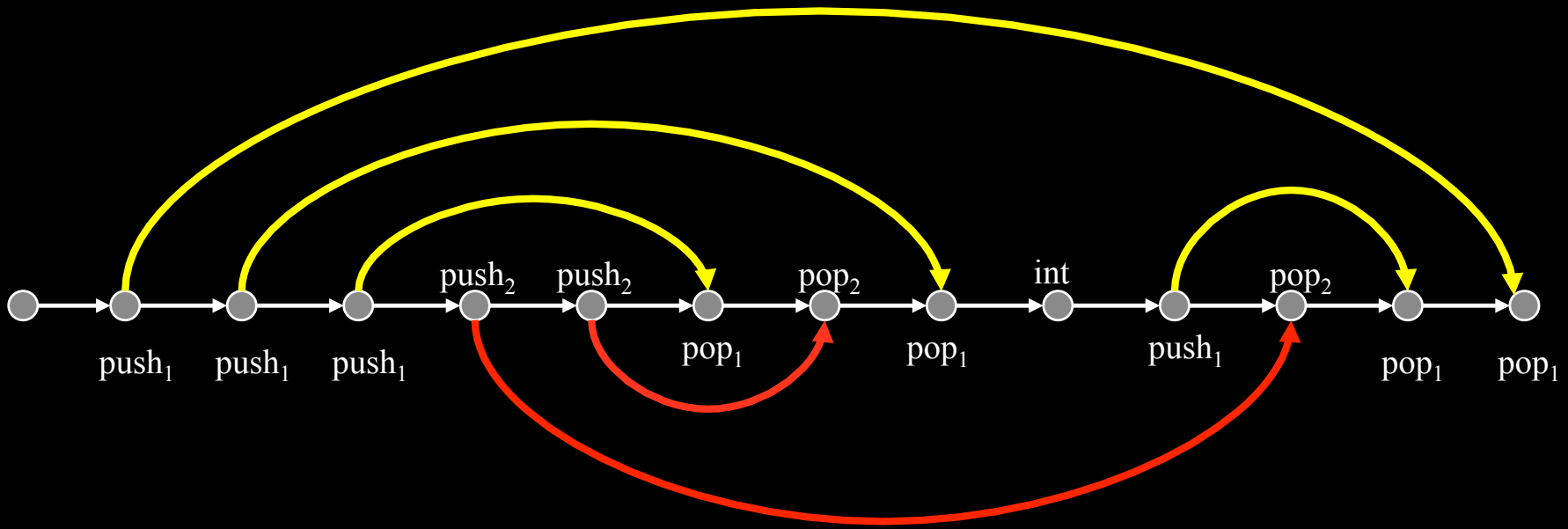




Hence the emptiness problem is decidable

Simulation
for
Multistack Pushdown Automata

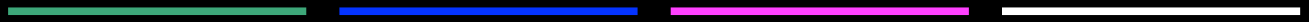
Multiply Nested Words (MNW)



- A MNW graph captures the behavior of a run
  - Stacks are compiled down into the graph (nesting edges)
- The class MNWs is MSO definable
- Unbounded tree-width
  - (undecidable emptiness problem)

Bounded-context MPAs

- k context-switches MPA (Qadeer, Rehof: TACAS'05)



In a context only one stack can be used

The class of MNWs with k contexts is

- MSO definable
- tree-width: k + 1

Hence emptiness problem is decidable.

Bounded-phase MPAs

- k-phase MPA (La Torre, Madhusudan, Parlato - LICS'07)

In a phase only one stack can be popped (all the stacks can be pushed)

MNWs with k phases are

- MSO definable
- tree-width: $2^k + 2^{k-1} + 1$

Hence emptiness problem is decidable.

Ordered MPAs

- Ordered MPAs (Breveglieri et al, *J. Found. Comput. Sci.'95*)

Only the first non empty stack can be popped
All stacks can be pushed

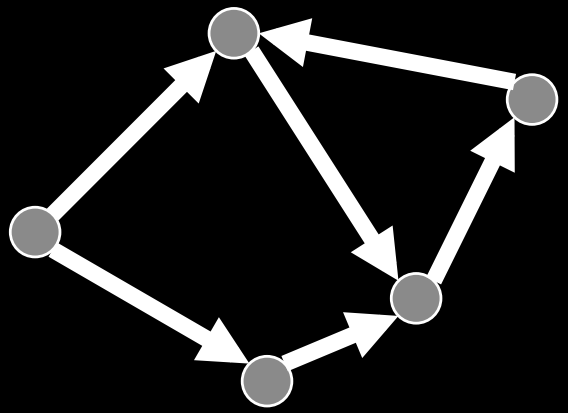
Ordered MNWs are
- MSO definable
- tree-width: $(n+1)\ 2^{n-1}+1$, where n is the number of stacks

Therefore emptiness problem is decidable.

Simulation
for
distributed automata
with queues

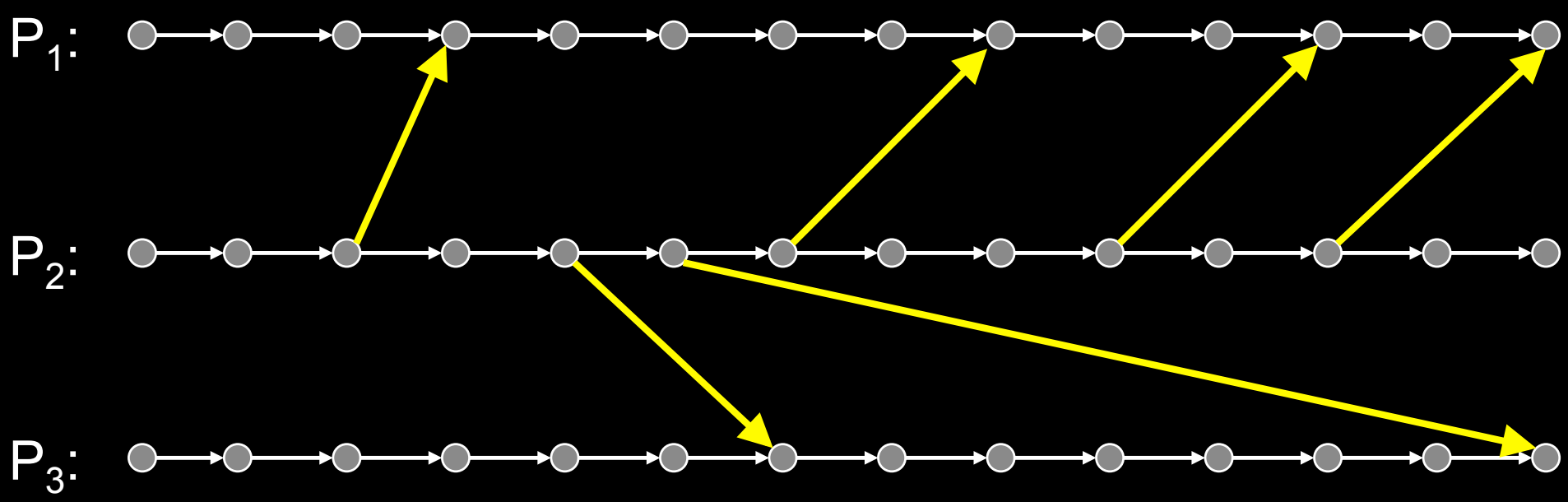
Distributed automata with queues

- Distributed automata with queues with
  - finite state processes
  - pushdown processes



Network of processes

Queue graphs (finite processes)

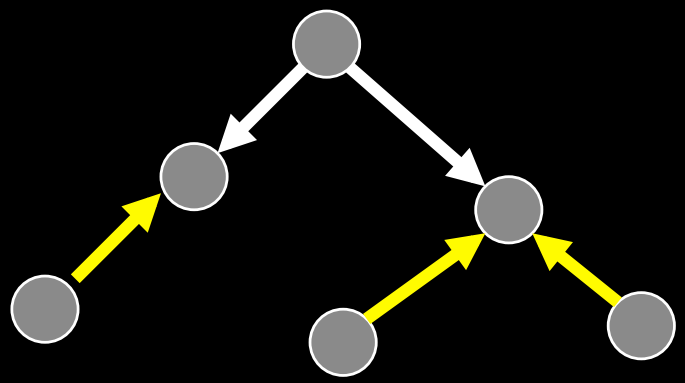


- A queue graph captures the behavior of a run
  - queues are compiled down into the graph (yellow edges)
- The class of queue graphs is MSO definable
  (linear orders for each process, FIFO edges)
- Unbounded tree-width

Decidable class (La Torre, Madhusudan, Parlato, TACAS'08)

- Finite processes - polyforest architectures

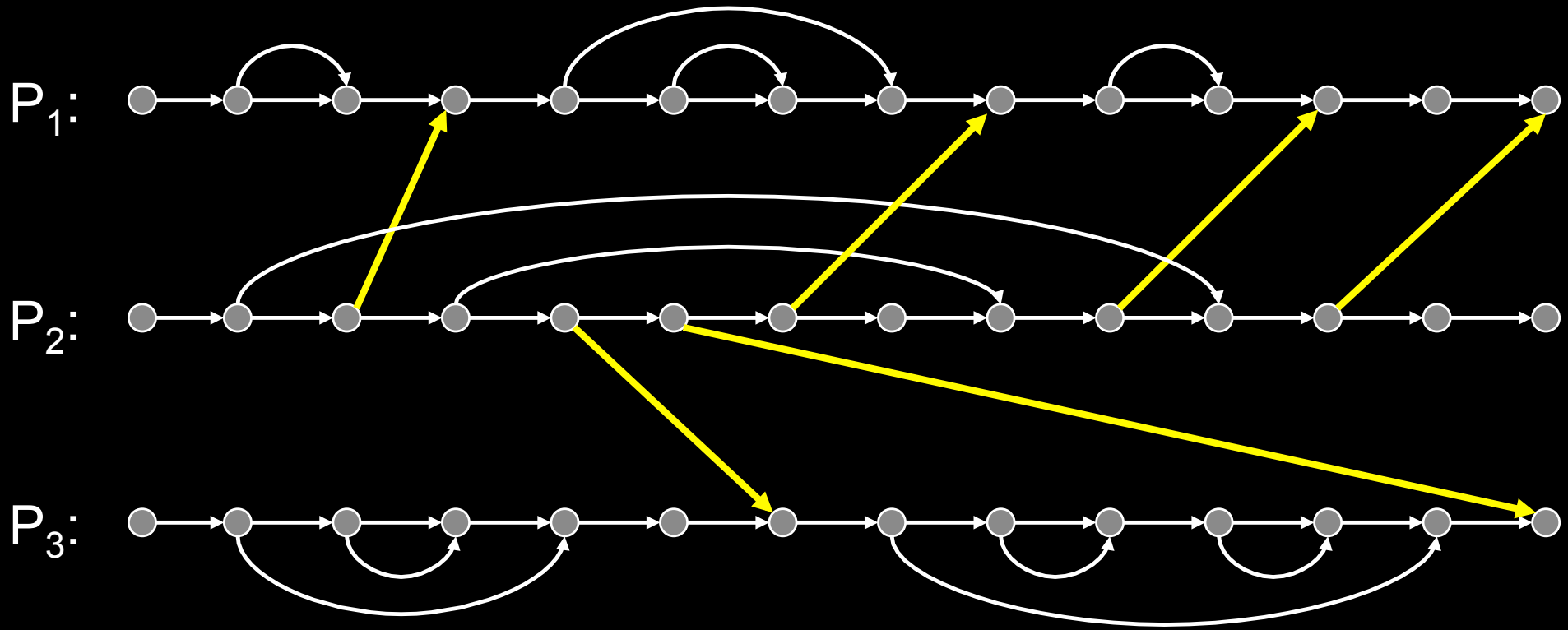
  A directed graph is a polyforest if the underlying undirected graph is a forest



TREE-WIDTH: 3n-1

Hence emptiness problem is decidable.

Stack-queue graphs (PD processes)



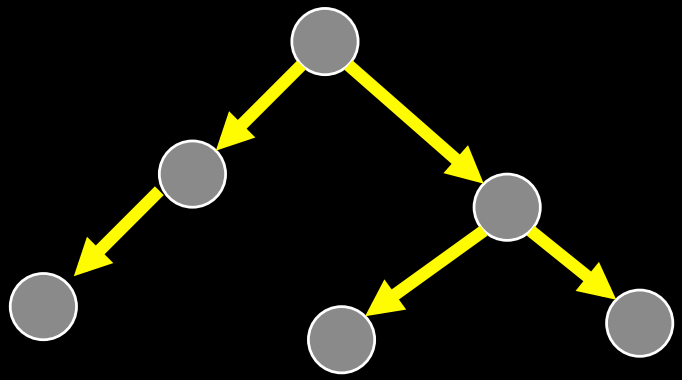
- A stack-queue graph captures the behavior of a run
  - Stacks and queues are compiled down into the graph
- The class stack-queue graphs is MSO definable
- Unbounded tree-width

Decidable class (La Torre, Madhusudan, Parlato, TACAS'08)

- Pushdown processes - directed forests + well-queuing

  Well-queuing condition:

  each process can receive a message only when its stack is empty



TREE-WIDTH: 3n-1

Hence emptiness problem is decidable.

Conclusions

We give simulations for …

- Pushdown automata
  - Behavior graphs: nested words
  - Tree-width: 2
- n-stack pushdown automata
  - Behavior graphs: n-nested words (unbounded tree-width)
  - Restrictions
    - k-contexts tree-width: $O(k)$
    - k-phases tree-width: $O(2^k)$
    - ordered tree-width: $O(n\, 2^n)$
- Distributed automata with n-processes & FIFO queues
  - Behavior graphs: stack-queue graphs (unbounded tree-width)
  - Restrictions:
    - finite state processes with polyforest architecture - tree-width: $O(n)$
    - pushdown processes with forest architecture + well queuing - tree-width: $O(n)$

Decidable emptiness problem

Multistack pushdown automata with

- k-contexts (Rehof, Qadeer - TACAS'05)
- k-phases (La Torre, Madhusudan, P. - LICS'07)
- ordered (Breveglieri, Cherubini, Citrini, Crespi-Reghizzi - *Int. J. Found. Comput. Sci.'95*)
- Parameterized pushdown automata with k-rounds (La Torre, Madhusudan, P. - CAV'10)

Distributed automata with FIFO queues

- finite state processes with polyforest architecture
- pushdown processes with forest architecture + well queuing (La Torre, Madhusudan, P. - TACAS'08)
- Non-confluent architectures + eager runs (Heußner, Leroux, Muscholl, Sutre - FOSSACS'10)

Decidable emptiness problem

Multistack pushdown automata with

- k-contexts
- k-phases
- ordered

S'05)

S'07)

95)

0)

A general criterion that uniformly explains many decidability results for automata with auxiliary storage

Complexity: matches best known time complexity

+ well queuing

(La Torre, Madhusudan, P. - TACAS'08)

+ eager runs

(Heußner, Leroux, Muscholl, Sutre - FOSSACS'10)

New results can be easily derived

- For multistack pushdown automata considering phases where in each phase only one stack can be pushed but all the stacks can be popped can be shown decidable
  - Flipping the direction of edges you get bounded-phase MNWs
- General result can be extended to graphs of bdd clique width
- Extends to infinite words
  - Eg. Buchi/parity ordered multi-stack automata is decidable
- ***A general Parikh theorem***

Take-home message

New view of automata with storage

- Don’t be “mesmerized” by the storage capabilities and restrictions
- Look instead at the the underlying graph that captures the storage

  (graph automaton working on these graphs must be able to simulate the automaton with aux storage)
- Look at the tree-width of this graph

Thank you

Future work

- Graph automata working over bounded tree-width graphs are a powerful class that can explain many emptiness results

  But what about complementation?

  Is there a “graph-theoretic” property that captures when automata can be complemented?

  (Bounded phase visibly pushdown languages are complementable!)

- We cannot handle counters effectively
  - Decision procedures for counters are very different (Petri-net coverability/ WQO)

  Can we incorporate this decision procedure also into a general theory?