Revisiting Underapproximate Reachability for Multipushdown Systems

Sparsa Roychowdhury, Prof. S Akshay, Prof. S Krishna, and Prof. Paul Gastin

July 26, 2024

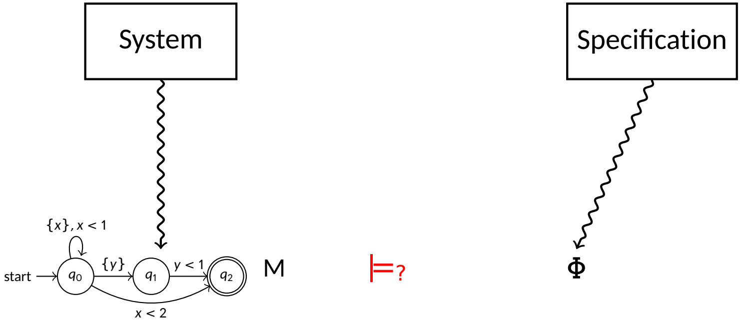
Outline

- Introduction
- Multistack Pushdown Automata (MPDA)
- Hole bounded runs of MPDA
- Algorithm
- Extension to time
- Implementation
- Experiments
- Conclusion

Introduction

- Checking the correctness of a system is essential
- Testing can guarantee the presence of bugs but can not guarantee the absence of it
- Formal verification is a celebrated area of computer science that proves the correctness of a system mathematically
- Model checking is one of the most promising approaches to formal verification

Model checking



$$L(M) \cap \overline{L(\Phi)} = \emptyset$$

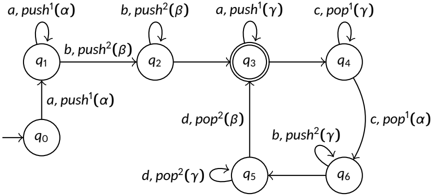
Figure: Model checking as a reachability/emptiness problem

Model checking contd.

- Finding a proper model is essential to capture the important behaviors of systems
- If the model is too powerful, model checking becomes difficult, sometimes undecidable!
- Thus, finding a good model that strikes a good balance between expressivity and complexity is very crucial in model checking

Multi-stack Pushdown Automata

- Verification of recursive concurrent programs is an active area of interest
- Behaviours of recursive concurrent programs can be modeled using multi-stack pushdown automata(MPDA)



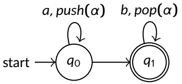
A Multistack Pushdown Automata accepting the language:
$L^{bh} = \{a^n b^n (a^{q_1} c^{q_1+1} b^{q'_1} d^{q'_1+1} \cdots a^{q_n} c^{q_n+1} b^{q'_n} d^{q'_n+1}) \mid n, q_i, q'_i \in \mathbb{N}, \forall i \in [n]\}$

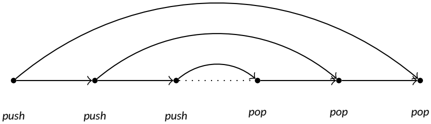
Multi-stack Pushdown Automata

- Multi-stack pushdown automata is Turing complete
- Reachability is Undecidable!!
- Under-approximation!
  1. Bounded Round La Torre et al. [2010], Qadeer and Wu [2004]
  2. Bounded Scope Torre et al. [2016]
  3. Bounded Phase La Torre et al. [2007] etc.

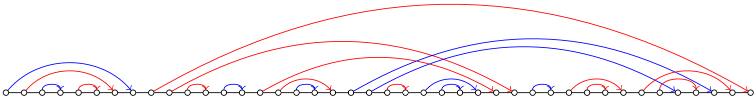
Capturing Behavior as Graphs

Graphs can be used to represent set of runs

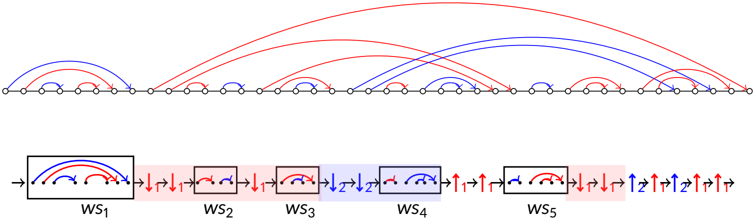




Behavior Graph of an MPDA

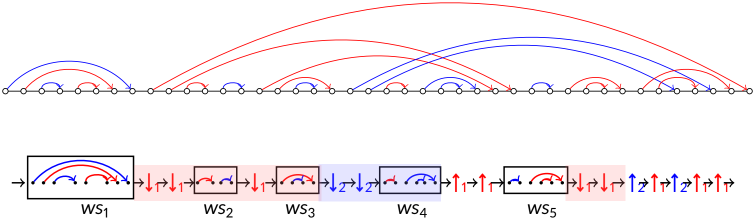


Behavior Graph of an MPDA



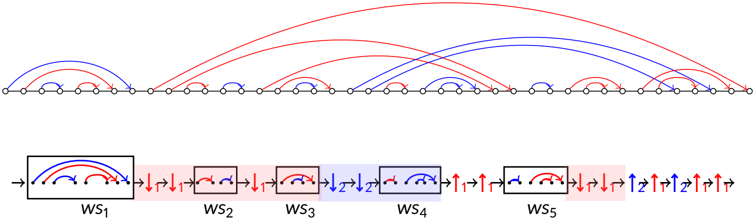
- Well-nested sequence (ws_j) can be ε

Behavior Graph of an MPDA



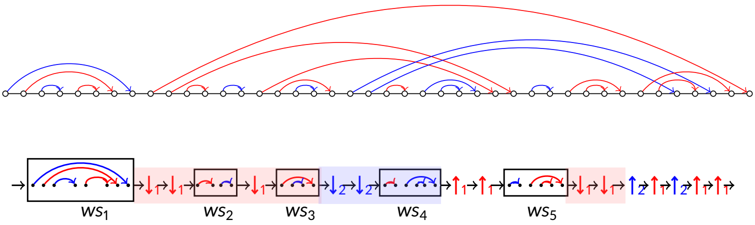
- Well-nested sequence (ws_j) can be ε
- Pushes not part of a well-nested sequence (represented as $\downarrow_i$)

Behavior Graph of an MPDA



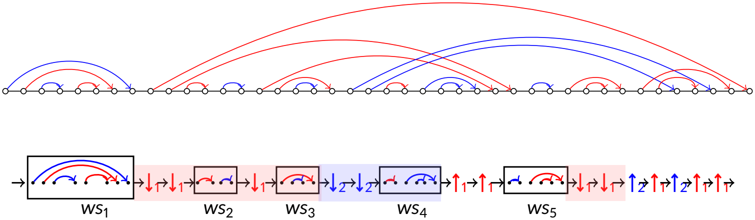
- Well-nested sequence (ws_j) can be ε
- Pushes not part of a well-nested sequence (represented as $\downarrow_i$)
- Matching pops that are matched with $\downarrow_i$ (represented as$\uparrow_i$)

Behavior Graph of an MPDA



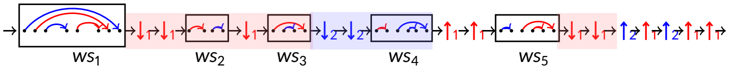
- Well-nested sequence (ws_j) can be ε
- Pushes not part of a well-nested sequence (represented as $\downarrow_i$)
- Matching pops that are matched with $\downarrow_i$ (represented as $\uparrow_i$)
- An atomic-i-segment of stack i is a push $\downarrow_i$ followed by a well-nested sequence ws_j.

Behavior Graph of an MPDA



- Well-nested sequence (ws_j) can be ε
- Pushes not part of a well-nested sequence (represented as $\downarrow_i$)
- Matching pops that are matched with $\downarrow_i$ (represented as$\uparrow_i$)
- An atomic-i-segment of stack i is a push $\downarrow_i$ followed by a well-nested sequence ws_j.
- An i-segment is the concatenation of multiple atomic-i-segments

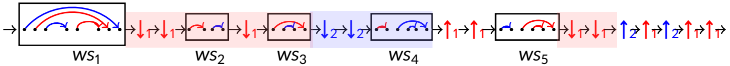
Segments and holes



- **Pending-push($\downarrow_i$)** of stack i (w.r.t a position n)

  `No matching pop prior` n

Segments and holes

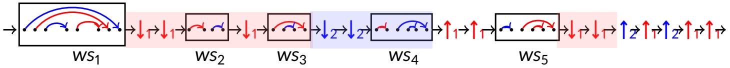


- **Pending-push($\downarrow_i$)** of stack i (w.r.t a position n)

  `No matching pop prior` n

- **An i-segment** is open if its first push transitions are pending

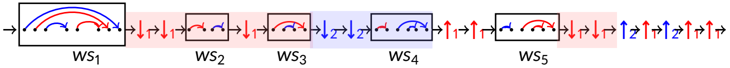
Segments and holes



- **Pending-push($\downarrow_i$)** of stack i (w.r.t a position n)

  `No matching pop prior n`
- **An i-segment** is open if its first push transitions are pending
- **An i-hole** in a run is the maximal factor that is an open i-segment

Segments and holes

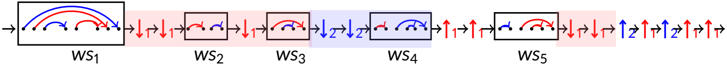


- **Pending-push($\downarrow_i$)** of stack i (w.r.t a position n)

  No matching pop prior n

- **An i-segment** is open if its first push transitions are pending
- **An i-hole** in a run is the maximal factor that is an open i-segment
- We denote by nbHoles(σ) the number of holes in a run σ

Hole Bounded runs of MPDA

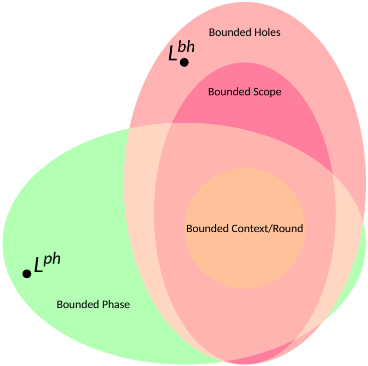


Definition

A run σ is said to be *K*-**hole** bounded if $\text{nbHoles}(\sigma) \leq K$, for $K \in \mathbb{N}$.

In **hole** bounded reachability of MPDA we only search for a **K-hole** bounded accepting run of MPDA where, **K**$\in \mathbb{N}$.

Hierarchy of Under-approximations



$$L^{ph} = \{(ab)^n c^n d^n | n \in \mathbb{N}\}$$

$$L^{bh} = \{a^n b^n (a^{q_1} c^{q_1+1} b^{q'_1} d^{q'_1+1} \cdots a^{q_n} c^{q_n+1} b^{q'_n} d^{q'_n+1}) \mid n, q_i, q'_i \in \mathbb{N}$$

Algorithm to Solve Hole bounded reachability

Question: Given a MPDA and a bound $K \in \mathbb{N}$ does there exists a K-hole bounded run that reaches a target state from the initial state?

- Algorithm
  - Uses fixed-point computation
  - Guided BFS exploration
  - Witness generation

Steps of the Algorithm

We divide the algorithm in two parts,

1. Fixed-point Computation and Memoization

   `Precomputes well nested runs (`ws`), atomic open-hole `$(\downarrow_i ws)$`, and open-hole `$(\downarrow_i ws)^+$`.`

Steps of the Algorithm

We divide the algorithm in two parts,

1. Fixed-point Computation and Memoization
2. On-The-Fly guided Breadth First Search (BFS) Exploration `Which solves hole bounded reachability for MPDA`

Computing Well-Nested Sequences

$$s_i \xrightarrow{\downarrow^n (\alpha)} s_j$$

Computing Well-Nested Sequences

$$\begin{tikzcd} s_i \arrow[r, "\downarrow^n (\alpha)"] & s_j \arrow[r] & s_k \end{tikzcd}$$

Computing Well-Nested Sequences

$$s_i \xrightarrow{\downarrow^n (\alpha)} s_j \longrightarrow s_k \xrightarrow{\uparrow^n (\alpha)} s_l$$

Computing Well-Nested Sequences

$$\begin{tikzcd} s_i \arrow[rr] \arrow[rr, bend left] & & s_l \end{tikzcd}$$

Computing Well-Nested Sequences

For all state s_i, create the following entries,

- (s_i, s_i)
- (s_i, s_k) if s_k is reachable by discrete transition

Computing Well-Nested Sequences

For all state s_i, create the following entries,

- (s_i, s_i)
- (s_i, s_k) if s_k is reachable by discrete transition

$$
\left.
\begin{array}{c|c|c|c|c}
\ddots & s_j & \ldots & s_k & \ldots \\
\hline
\vdots & \vdots & \ldots & \vdots & \ldots \\
\hline
s_i & 1 & \ldots & 0 & \ldots \\
\hline
\vdots & \vdots & \ldots & \vdots & \ddots
\end{array}
\right\} \mid \mathscr{S} \mid
$$

Computing Well-Nested Sequences

For all state s_i, create the following entries,

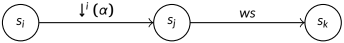
- (s_i, s_i)
- (s_i, s_k) if s_k is reachable by discrete transition

$$\left.\begin{array}{c|c|c|c|c} \ddots & s_j & \dots & s_k & \dots \\ \hline \vdots & \vdots & \dots & \vdots & \dots \\ \hline s_i & 1 & \dots & 0 & \dots \\ \hline \vdots & \vdots & \dots & \vdots & \ddots \end{array}\right\} |\mathscr{S}|$$

Transitive Closure will give us the **well-nested** sequences

Computing i-segments and i-holes

- An **i-segment ($\downarrow^i (\alpha) ws$)** for a stack i is a pending push($\downarrow^i$) followed by a maximal **well-nested** sequence(ws)



Computing i-segments and i-holes

- An **i-segment ($\downarrow^i(\alpha)ws$)** for a stack i is a pending push($\downarrow^i$) followed by a maximal **well-nested** sequence(ws)
- We can pre-compute all possible **i-segments** by concatenating well-nested sequences after a push transition of stack i.

$$s_i \xrightarrow{\downarrow^i(\alpha)} s_j \xrightarrow{ws} s_k$$

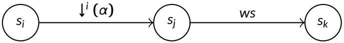
Computing i-segments and i-holes

- An **i-segment ($\downarrow^i$ $(\alpha)ws$)** for a stack i is a pending push($\downarrow^i$) followed by a maximal **well-nested** sequence(ws)
- We can pre-compute all possible **i-segments** by concatenating well-nested sequences after a push transition of stack i.
- Assume we store this information in M^i

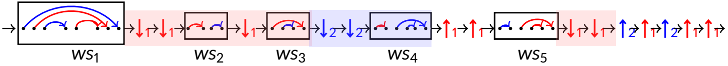
$$
\begin{tikzcd}
s_i \arrow[r, "\downarrow^i (\alpha)"] & s_j \arrow[r, "ws"] & s_k
\end{tikzcd}
$$

Computing i-segments and i-holes

- An **i-segment ($\downarrow^i (\alpha) ws$)** for a stack i is a pending push($\downarrow^i$) followed by a maximal **well-nested** sequence(ws)
- We can pre-compute all possible **i-segments** by concatenating well-nested sequences after a push transition of stack i.
- Assume we store this information in M^i
- Dropping stack alphabet and doing transitive closure on M^i gives us information of all **i-Holes**, (M_i^h).



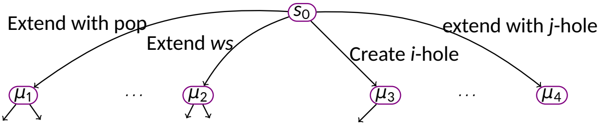
Blocks of a run of MPDA



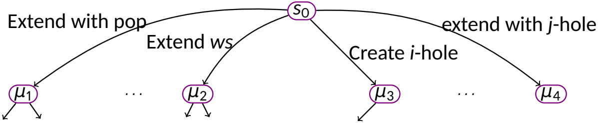
Every run of a MPDA is made of three basic blocks

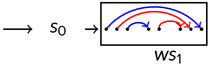
- Well-nested sequence (ws_j)
- Hole (sequence of $\downarrow_i$ ws_j)
- Matching-pops ($\uparrow_i$)

Breadth First Search Exploration

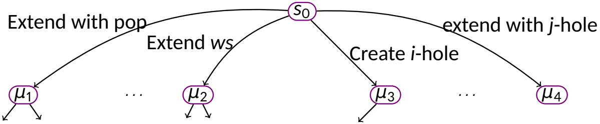


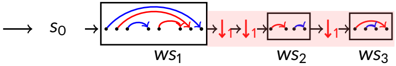
Breadth First Search Exploration



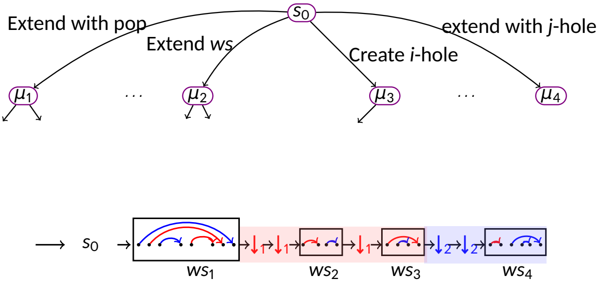


Breadth First Search Exploration

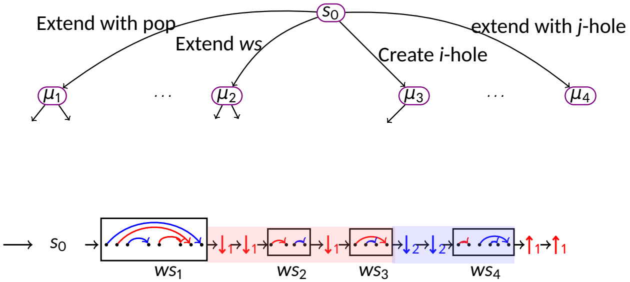




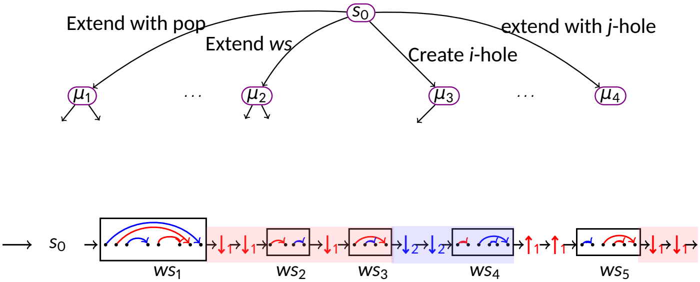
Breadth First Search Exploration



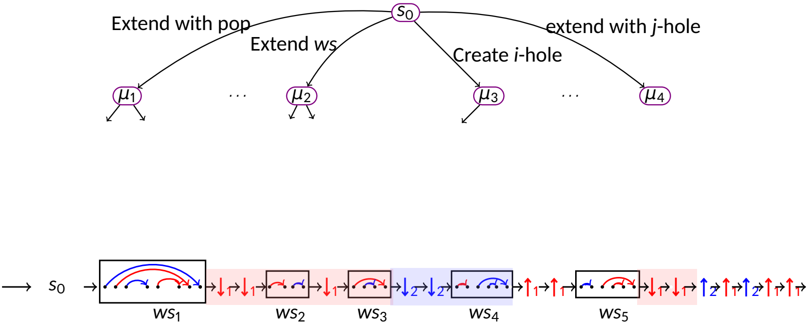
Breadth First Search Exploration



Breadth First Search Exploration



Breadth First Search Exploration



Witness

For well-nested runs, we can generate witness by a recursive procedure.

Witness

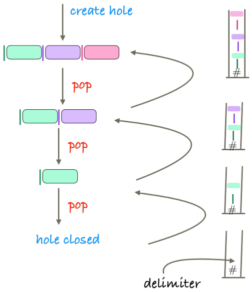
For well-nested runs, we can generate witness by a recursive procedure.

For runs which are not well-nested backtracking is used

Witness

For well-nested runs, we can generate witness by a recursive procedure.

For runs which are not well-nested backtracking is used



Timed Automata Alur and Dill [1994]

Timed automaton is a well-studied model that capture the behaviors of real-time systems

- Finite States + Clocks

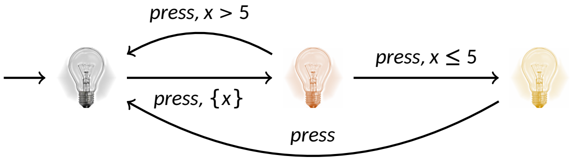


Figure: A Timed Automaton.

- Infinite state space
- Region construction
- Reachability is PSPACE-C

Extension to Time

We also extend the algorithm to handle Timed Multistack Pushdown Automata

- Clocks + Age (closed guards)
- Data Structures
- Timed witness generation

BHIM

- We have built a tool BHIM[1] based on the algorithm
- Works on both timed and untimed MPDA
- Generates a run as a witness of reachability
- We tested our tool on a set of benchmarks

[1] https://sparsa.github.io/bhim

Experiments

Name	Locations	Transitions	Stacks	Holes	Time Empty (mili sec)	Time Witness (mili sec)	Memory(KB)
Bluetooth	45	89	2	0	149.3	0.241	6934
MultiProdCons-1(2)	11	18	2	2	11.1	0.1	1796
MultiProdCons-2(3,2)	7	11	2	2	126.529	0.281	5632
MultiProdCons-2(24,7)	32	34	2	2	1879.33	10.63	21836
dm-target	22	27	2	2	26.483	0.279	6624
Binary Search Tree	29	78	2	2	60.8	5.1	5143
untimed-L^{crit}	6	10	2	2	14.9	0.7	4692
Untimed-Maze	9	12	2	0	8.25	0.07	5558
L^{bh}	7	13	2	2	22.2	0.6	4404

Table: Experimental results: Time Empty and Time Witness column represents no. of milliseconds needed for emptiness checking and to generate witness, respectively.

Name	Locations	Transitions	Stacks	Clocks	c_{max}	Aged(Y/N)	Holes	Time Empty(mili sec)	Time Witness (mili sec)	Memory(KB)
Bluetooth	45	89	2	0	2	Y	0	152.8	0.119	5568
L^{crit}	6	10	2	2	8	Y	2	9965.2	3.7	203396
Maze	9	12	2	2	5	Y	2	349.3	0.31	11604

Table: Experimental results of timed examples. The column c_{max} is defined as the maximum constant in the automaton, and Aged denotes if the stack is timed or not

Conclusion

- A new underapproximation

Conclusion

- A new underapproximation
- Subsumes already known underapproximations

Conclusion

- A new underapproximation
- Subsumes already known underapproximations
- Decidable reachability

Conclusion

- A new underapproximation
- Subsumes already known underapproximations
- Decidable reachability
- BHIM

Thank You

Thank You

Questions?

Bibliography I

S. Akshay, Paul Gastin, Shankara Narayanan Krishna, and Ilias Sarkar. Towards an efficient tree automata based technique for timed systems. In *28th International Conference on Concurrency Theory, CONCUR 2017, September 5-8, 2017, Berlin, Germany*, pages 39:1–39:15, 2017. URL https://doi.org/10.4230/LIPIcs.CONCUR.2017.39.

Rajeev Alur and David L Dill. A theory of timed automata. *Theoretical computer science*, 126(2):183–235, 1994.

Salvatore La Torre, Parthasarathy Madhusudan, and Gennaro Parlato. A robust class of context-sensitive languages. In *Logic in Computer Science, 2007. LICS 2007. 22nd Annual IEEE Symposium on*, pages 161–170. IEEE, 2007.

Salvatore La Torre, Parthasarathy Madhusudan, and Gennaro Parlato. The language theory of bounded context-switching. In *Latin American Symposium on Theoretical Informatics*, pages 96–107. Springer, 2010.

P Madhusudan and Gennaro Parlato. The tree width of auxiliary storage. In *ACM SIGPLAN Notices*, volume 46, pages 283–294. ACM, 2011.

Shaz Qadeer and Dinghao Wu. Kiss: keep it simple and sequential. *ACM sigplan notices*, 39(6):14–24, 2004.

Salvatore La Torre, Margherita Napoli, and Gennaro Parlato. Scope-bounded pushdown languages. *International Journal of Foundations of Computer Science*, 27(02):215–233, 2016.

Decidability & Some Possible Algorithms

Theorem

The K-hole bounded reachability of MPDA *is decidable*

- Hole bounded ⇒ Tree-width bounded

Decidability & Some Possible Algorithms

Theorem

The K-hole bounded reachability of MPDA *is decidable*

- Hole bounded ⇒ Tree-width bounded
- Tree-width bounded ⇒ Decidable reachability Madhusudan and Parlato [2011]

Decidability & Some Possible Algorithms

Theorem

The K-hole bounded reachability of MPDA *is decidable*

- Hole bounded $\Rightarrow$ Tree-width bounded
- Tree-width bounded $\Rightarrow$ Decidable reachability Madhusudan and Parlato [2011]

Decidability & Some Possible Algorithms

Theorem

The K-hole bounded reachability of MPDA *is decidable*

- Hole bounded ⇒ Tree-width bounded
- Tree-width bounded ⇒ Decidable reachability Madhusudan and Parlato [2011]
- Tree-width based Algorithm Akshay et al. [2017],

Decidability & Some Possible Algorithms

Theorem

The K-hole bounded reachability of MPDA *is decidable*

- Hole bounded ⇒ Tree-width bounded
- Tree-width bounded ⇒ Decidable reachability Madhusudan and Parlato [2011]
- Tree-width based Algorithm Akshay et al. [2017],
  1. Capture the behaviour of the model as a graph

Decidability & Some Possible Algorithms

Theorem

The K-hole bounded reachability of MPDA *is decidable*

- Hole bounded $\Rightarrow$ Tree-width bounded
- Tree-width bounded $\Rightarrow$ Decidable reachability Madhusudan and Parlato [2011]
- Tree-width based Algorithm Akshay et al. [2017],
  1. Capture the behaviour of the model as a graph
  2. Show that these graphs has bounded tree-width

Decidability & Some Possible Algorithms

Theorem

The K-hole bounded reachability of MPDA *is decidable*

- Hole bounded ⇒ Tree-width bounded
- Tree-width bounded ⇒ Decidable reachability Madhusudan and Parlato [2011]
- Tree-width based Algorithm Akshay et al. [2017],
  1. Capture the behaviour of the model as a graph
  2. Show that these graphs has bounded tree-width
  3. Solve Reachability using Tree-automata

Decidability & Some Possible Algorithms

Theorem

The K-hole bounded reachability of MPDA *is decidable*

- Hole bounded ⇒ Tree-width bounded
- Tree-width bounded ⇒ Decidable reachability Madhusudan and Parlato [2011]
- Tree-width based Algorithm Akshay et al. [2017],
  1. Capture the behaviour of the model as a graph
  2. Show that these graphs has bounded tree-width
  3. Solve Reachability using Tree-automata

Decidability & Some Possible Algorithms

Theorem

The K-hole bounded reachability of MPDA *is decidable*

- Hole bounded ⇒ Tree-width bounded
- Tree-width bounded ⇒ Decidable reachability Madhusudan and Parlato [2011]
- Tree-width based Algorithm Akshay et al. [2017],
  1. Capture the behaviour of the model as a graph
  2. Show that these graphs has bounded tree-width
  3. Solve Reachability using Tree-automata

Difficult to design!

- Naive BFS exploration

Decidability & Some Possible Algorithms

Theorem

The K-hole bounded reachability of MPDA *is decidable*

- Hole bounded ⇒ Tree-width bounded
- Tree-width bounded ⇒ Decidable reachability Madhusudan and Parlato [2011]
- Tree-width based Algorithm Akshay et al. [2017],
  1. Capture the behaviour of the model as a graph
  2. Show that these graphs has bounded tree-width
  3. Solve Reachability using Tree-automata

Difficult to design!

- Naive BFS exploration
  1. makes arbitrary nondeterministic choices

Decidability & Some Possible Algorithms

Theorem

The K-hole bounded reachability of MPDA *is decidable*

- Hole bounded ⇒ Tree-width bounded
- Tree-width bounded ⇒ Decidable reachability Madhusudan and Parlato [2011]
- Tree-width based Algorithm Akshay et al. [2017],
  1. Capture the behaviour of the model as a graph
  2. Show that these graphs has bounded tree-width
  3. Solve Reachability using Tree-automata

Difficult to design!

- Naive BFS exploration
  1. makes arbitrary nondeterministic choices

Decidability & Some Possible Algorithms

Theorem

The K-hole bounded reachability of MPDA *is decidable*

- Hole bounded ⇒ Tree-width bounded
- Tree-width bounded ⇒ Decidable reachability Madhusudan and Parlato [2011]
- Tree-width based Algorithm Akshay et al. [2017],
  1. Capture the behaviour of the model as a graph
  2. Show that these graphs has bounded tree-width
  3. Solve Reachability using Tree-automata

Difficult to design!

- Naive BFS exploration
  1. makes arbitrary nondeterministic choices

Unnecessary branches!