C. AISWARYA, PAUL GASTIN, PRAKASH SAIVASAN

ORDER-2 NESTED WORDS
HIGHER ORDER PUSHDOWN SYSTEMS

- Pushdown (stack)
  - pop, push(a), ?top(a)

lababa...babb
PROLOGUE

HIGHER ORDER PUSHDOWN SYSTEMS

- Pushdown (stack)
  - pop, push(a), ?top(a)
PROLOGUE

HIGHER ORDER PUSHDOWN SYSTEMS

- Pushdown (stack)

  - pop, push(a), ?top(a)

- \[ \text{\underline{ababa}} \ldots \text{\underline{baba}} \]
HIGHER ORDER PUSHDOWN SYSTEMS

- order-2 stack (stack of stacks)
  - pop, push(a), ?top(a)
  - Push2, Pop2
HIGHER ORDER PUSHDOWN SYSTEMS

- order-2 stack (stack of stacks)
  - pop, push(a), ?top(a)
  - Push2, Pop2

\[
\downarrow a...ba \downarrow b...bb \quad \cdots \quad \downarrow ababa...baba \downarrow ababa...baba
\]
PROLOGUE

HIGHER ORDER PUSHDOWN SYSTEMS

- order-2 stack (stack of stacks)
  - pop, push(a), ?top(a)
  - Push2, Pop2
PROLOGUE

HIGHER ORDER PUSHDOWN SYSTEMS

- order-2 stack (stack of stacks)
  - pop, push(b), ?top(a)
  - Push2, Pop2
  - Collapse

push(a)
PROLOGUE

HIGHER ORDER PUSHDOWN SYSTEMS

- order-2 stack (stack of stacks)
  - pop, push(b), ?top(a)
  - Push2, Pop2
  - Collapse

push(a)
HIGHER ORDER PUSHDOWN SYSTEMS

- order-2 stack (stack of stacks)
  - pop, push(b), ?top(a)
  - Push2, Pop2
  - Collapse

push(a) Push2
HIGHER ORDER PUSHDOWN SYSTEMS

- order-2 stack (stack of stacks)
  - pop, push(b), ?top(a)
  - Push2, Pop2
  - Collapse

```
push(a)Push2
```
HIGHER ORDER PUSHDOWN SYSTEMS

- order-2 stack (stack of stacks)
  - pop, push(b), ?top(a)
  - Push2, Pop2
  - Collapse

\[\text{push}(a) \text{Push2} \text{push}(b)\]
HIGHER ORDER PUSHDOWN SYSTEMS

- order-2 stack (stack of stacks)
  - pop, push(b), ?top(a)
  - Push2, Pop2
  - Collapse

push(a)Push2push(b)
HIGHER ORDER PUSHDOWN SYSTEMS

- order-2 stack (stack of stacks)
  - pop, push(b), ?top(a)
  - Push2, Pop2
  - Collapse

push(a)Push2push(b)Push2
HIGHER ORDER PUSHDOWN SYSTEMS

- order-2 stack (stack of stacks)
  - pop, push(b), ?top(a)
  - Push2, Pop2
  - Collapse

```
push(a) Push2 push(b) Push2
```
HIGHER ORDER PUSHDOWN SYSTEMS

- order-2 stack (stack of stacks)
  - pop, push(b), ?top(a)
  - Push2, Pop2
  - Collapse

push(a)Push2push(b)Push2push(a)
HIGHER ORDER PUSHDOWN SYSTEMS

- order-2 stack (stack of stacks)
  - pop, push(b), ?top(a)
  - Push2, Pop2
  - Collapse

\[
push(a) Push2 push(b) Push2 push(a)
\]
HIGHER ORDER PUSHDOWN SYSTEMS

- order-2 stack (stack of stacks)
  - pop, push(b), ?top(a)
  - Push2, Pop2
  - Collapse

push(a) Push2 push(b) Push2 push(a) Push2
HIGHER ORDER PUSHDOWN SYSTEMS

- order-2 stack (stack of stacks)
  - pop, push(b), ?top(a)
  - Push2, Pop2
  - Collapse

push(a)Push2push(b)Push2push(a)Push2
HIGHER ORDER PUSHDOWN SYSTEMS

- order-2 stack (stack of stacks)
  - pop, push(b), ?top(a)
  - Push2, Pop2
  - Collapse

push(a)Push2push(b)Push2push(a)Push2Collapse
HIGHER ORDER PUSHDOWN SYSTEMS

- order-2 stack (stack of stacks)
  - pop, push(b), ?top(a)
  - Push2, Pop2
  - Collapse
HIGHER ORDER PUSHDOWN SYSTEMS

- order-2 stack (stack of stacks)
  - pop, push(b), ?top(a)
  - Push2, Pop2
  - Collapse

push(a)Push2push(b)Push2push(a)Push2
HIGHER ORDER PUSHDOWN SYSTEMS

- order-2 stack (stack of stacks)
  - pop, push(b), ?top(a)
  - Push2, Pop2
  - Collapse

```
push(a) Push2 push(b) Push2 push(a) Push2 Pop2
```
HIGHER ORDER PUSHDOWN SYSTEMS

- order-2 stack (stack of stacks)
  - pop, push(b), ?top(a)
  - Push2, Pop2
  - Collapse

push(a)Push2push(b)Push2push(a)Push2Pop2
HIGHER ORDER PUSHDOWN SYSTEMS

- order-2 stack (stack of stacks)
  - pop, push(b), ?top(a)
  - Push2, Pop2
  - Collapse

push(a) Push2 push(b) Push2 push(a) Push2
HIGHER ORDER PUSHDOWN SYSTEMS

- order-2 stack (stack of stacks)
  - pop, push(b), ?top(a)
  - Push2, Pop2
  - Collapse

push(a)Push2push(b)Push2push(a)Push2pop
HIGHER ORDER PUSHDOWN SYSTEMS

- order-2 stack (stack of stacks)
  - pop, push(b), ?top(a)
  - Push2, Pop2
  - Collapse

push(a)Push2push(b)Push2push(a)Push2pop
HIGHER ORDER PUSHDOWN SYSTEMS

- order-2 stack (stack of stacks)
  - pop, push(b), ?top(a)
  - Push2, Pop2
  - Collapse

push(a)Push2push(b)Push2push(a)Push2popCollapse
HIGHER ORDER PUSHDOWN SYSTEMS

- order-2 stack (stack of stacks)
  - pop, push(b), ?top(a)
  - Push2, Pop2
  - Collapse

push(a)Push2push(b)Push2push(a)Push2popCollapse
OUTLINE

HIGHER ORDER PUSHDOWN SYSTEMS (OF ORDER 2)

- prologue
- 2CPDS - a brief overview of results
- 2NW: a different way of understanding CPDS behaviours
- Nestify
- Spec languages: MSO, PDL, - UNDECIDABALITY
- under-approximations - tree interpretations - decidability
ORDER-2 COLLAPSIBLE PUSHDOWN SYSTEMS

- finite state machine equipped with an order-2 stack
- transitions perform order-2 stack operations
2-CPDS - BRIEF OVERVIEW OF RESULTS

ORDER-2 COLLAPSIBLE PUSHDOWN SYSTEMS

- 2-CPDS capture order-2 recursion schemes (Hague et al 2008)
- 2-CPDS strictly more powerful than 2-PDS for rec. schemes (Parys 2012)
- thought of as generating words, trees or graphs
- $\mu$-calculus model checking of 2-CPDS-trees is decidable (Ong 2006)
  - via parity games on configuration graphs of CPDS (Hague et al 2008)
  - via Krivine machines (SalvatiWalukiewicz 2011)
- configuration graphs of 2-CPDS can encode semi grids (Hague et al 2008)
- $\mu$-calculus over configuration graphs of 2-PDS decidable
OUTLINE

HIGHER ORDER PUSHDOWN SYSTEMS (OF ORDER 2)

- prologue
- 2CPDS - a brief overview of results
- 2NW: a different way of understanding CPDS behaviours
- Nestify
- Spec languages: MSO, PDL, - UNDECIDABALITY
- under-approximations - tree interpretations - decidability
ORDER-2 NESTED WORDS

NESTED WORDS

push(a)
ORDER-2 NESTED WORDS

NESTED WORDS

push(a)
ORDER-2 NESTED WORDS

NESTED WORDS

push(a) Push2
ORDER-2 NESTED WORDS

NESTED WORDS

push(a)Push2
ORDER-2 NESTED WORDS

NESTED WORDS

push(a) Push2 push(b)
ORDER-2 NESTED WORDS

NESTED WORDS

push(a) Push2 push(b)
push(a) Push2 push(b) Push2
ORDER-2 NESTED WORDS

NESTED WORDS

push(a) Push2 push(b) Push2
ORDER-2 NESTED WORDS

NESTED WORDS

push(a) Push2 push(b) Push2 push(a)
ORDER-2 NESTED WORDS

NESTED WORDS

push(a) Push2 push(b) Push2 push(a)
ORDER-2 NESTED WORDS

NESTED WORDS

push(a) Push2 push(b) Push2 push(a) Push2
ORDER-2 NESTED WORDS

NESTED WORDS

push(a) Push2 push(b) Push2 push(a) Push2
ORDER-2 NESTED WORDS

NESTED WORDS

push(a) Push2 push(b) Push2 push(a) Push2 Pop2
ORDER-2 NESTED WORDS

NESTED WORDS

push(a) Push2 push(b) Push2 push(a) Push2 Pop2
ORDER-2 NESTED WORDS

NESTED WORDS

push(a) Push2 push(b) Push2 push(a) Push2
ORDER-2 NESTED WORDS

NESTED WORDS

push(a) Push2 push(b) Push2 push(a) Push2 pop
ORDER-2 NESTED WORDS

NESTED WORDS

push(a) Push2 push(b) Push2 push(a) Push2 pop
NESTED WORDS

push(a) Push2 push(b) Push2 push(a) Push2 pop Collapse
ORDER-2 NESTED WORDS

NESTED WORDS

push(a) Push2 push(b) Push2 push(a) Push2 pop Collapse
ORDER-2 NESTED WORDS

EXAMPLES

1 — 2 — 3 — 4 — 5 — 6 — 7 — 8 — 9 — 10

↑₁ ↑₁ ↑₂ ↓₁ ↑₂ ↓₁ ↓₂ ↓₂ ↓₁ ↓₁
ORDER-2 NESTED WORDS

EXAMPLES

```
 1 — 2 — 3 — 4 — 5 — 6 — 7 — 8 — 9 — 10
  ↑1  ↑1  ↑2  ↓1  ↑2  ↓1  ↓2  ↓2  ↓1  ↓1
```

```
  ↑1  ↑1  ↑2  ↓1  ↓1  ↑2  ↓1  ↑1  ↑2  ↓1  ↓2  ↓1  ↓2  ↓1  ↓1  ↑1  ↑2  ↓1  ↓2  ↓1
```
ORDER-2 NESTED WORDS

LANGUAGE OF ORDER-2 CPDS

\[ L(\ldots) = \begin{cases} \end{cases} \]
MOTIVATION FOR STUDYING NESTED WORDS

WHAT ARE ORDER-2 NESTED WORDS?

- Not words
- Not trees
- Not configuration graphs
- Linear time behaviours extra information (matching)
  - Words with binary relation on its positions = special graphs
MOTIVATION FOR STUDYING NESTED WORDS

WHY ARE ORDER-2 NESTED WORDS INTERESTING?

- They make visible the data flow / communication
- Making visible the data flow is not new.
  - Widely used in message passing distributed systems (Message Sequence Charts - MSC)
  - also in pushdown and multi-pushdown systems (Nested words, multiply-nested words)
MOTIVATION FOR STUDYING NESTED WORDS

MESSAGE PASSING DISTRIBUTED SYSTEMS

Message Sequence Charts
ITU Standard
MOTIVATION FOR STUDYING NESTED WORDS

RECURSIVE PROGRAMS / XML

Nested Words
Alur, Madhusudan, 2009
MOTIVATION FOR STUDYING NESTED WORDS

MULTI-THREADED RECURSIVE PROGRAMS

Multiply Nested Words
MOTIVATION FOR STUDYING NESTED WORDS

RECURSIVE PROCESSES COMMUNICATING VIA MESSAGE PASSING

Concurrent Behaviours with Matching
WHY ARE ORDER-2 NESTED WORDS INTERESTING?

- They make visible the data flow / communication
- Making visible the data flow is not new.
  - Widely used in message passing distributed systems (Message Sequence Charts -MSC)
  - also in pushdown and multi-pushdown systems (Nested words, multiply-nested words)
WHY NESTED WORDS

- yet another way of understanding the model
- for better specification
  - more powerful specification formalism
  - can make use of the nesting edges when expressing properties
- verification beyond reachability
Letter before outermost call is the same as the letter after its return
∀x, y (a(x − 1) ∧ x ▷ y ∧ ¬∃z, z′ (z ▷ z′ ∧ z < x < z′)) ⇒ a(y + 1)

Letter before outermost call is the same as the letter after its return
Relate outermost call and returns

Not expressible in MSO over Linear Traces even with visible alphabet.
MOTIVATION FOR STUDYING NESTED WORDS

WHAT CAN ORDER-2 NESTED WORDS MODEL?

- Data-flow-visible linear behaviors of order-2 CPDS
- Can model recursive program with stack data-structure, where stacks can be passed to subroutines (pass-by-value)
MOTIVATION FOR STUDYING NESTED WORDS

WHAT CAN ORDER-2 NESTED WORDS MODEL?

- Data flow visible linear behaviors of order-2 CPS
- Can model recursive program with stack data-structure, where stacks can be passed to subroutines (pass-by-value)
- Can capture branching behaviour of pushdown systems (similar to nested trees (AlurChaudhuriMadhusudhan’06))
OUTLINE

HIGHER ORDER PUSHDOWN SYSTEMS (OF ORDER 2)

- prologue
- 2CPDS - a brief overview of results
- 2NW: a different way of understanding CPDS behaviours
- Nestify
- Spec languages: MSO, PDL, - UNDECIDABALITY
- under-approximations - tree interpretations - decidability
Words + two binary relations. When is it a valid 2-NW?
VALID ORDER-2 NESTED WORDS

- Words + two binary relations. When is it a valid 2-NW?
- Given a sequence of order-2 stack operations is it valid?
- Ops = \{ top=a?, top=b?, push(a), push(b), pop(a), pop(b), DupStack, PopStack, Collapse \}
- Naïvely simulate an order-2 stack
  - May take exponential space/time
VALID ORDER-2 NESTED WORDS

- Words + two binary relations. When is it a valid 2-NW?
- Given a sequence of order-2 stack operations is it valid?
- Can we construct the nesting relations from such a sequence?
- Nestify does it in linear time/space

push1'(x) = push1(x-1)

VALID ORDER-2 NESTED WORDS

- based on the inductive characterisation.
- push1(i), Push2(i)

\[
push_1(j) = \begin{cases} 
  j & \text{if } \text{op}_j \in \{\uparrow_s \mid s \in S\} \\
  push_1'(j) & \text{if } \text{op}_j \in \{\text{nop}, \uparrow_2\} \cup \{s? \mid s \in S\} \\
  push_1'(push_1'(j)) & \text{if } \text{op}_j = \downarrow_1 \\
  push_1'(Push_2'(j)) & \text{if } \text{op}_j = \downarrow_2 \\
  push_1'(Push_2'(push_1'(j))) & \text{if } \text{op}_j = \downarrow \\
\end{cases}
\]
CONFORM AND RELATIONS CONFORMING TO A VALID OPERATION SEQUENCE.

This is implemented in the tool.

Let us explain the last condition. Assume that be a valid operation sequence. We say that two binary relations

Proof.

C. Aiswarya, P. Gastin, and P. Saivasan 5

Having defined the functions is exactly the order-2 stack before points to the order-1 stack just below the top order-1 stack before (or after) push. Moreover, the top element after push Assume push that the top element after push, which was pushed at push. The cases

Push2(j) = \begin{cases} 
  j & \text{if } op_j = \uparrow_2 \\
  Push_2(j) & \text{if } op_j \in \{nop, \downarrow_1\} \cup \{\uparrow_1^s, \downarrow s? | s \in S\} \\
  Push_2(Push_2'(j)) & \text{if } op_j = \downarrow_2 \\
  Push_2(Push_2(Push_1'(j))) & \text{if } op_j = \downarrow \\
\end{cases}

The proof is by induction on push.
VALID ORDER-2 NESTED WORDS

- Words + two binary relations. When is it a valid 2-NW?
- Given a sequence of order-2 stack operations is it valid?
- Can we construct the nesting relations from such a sequence?
- Nestify does it in linear time/space
- keeps an array with two pointers

OUTLINE

HIGHER ORDER PUSHDOWN SYSTEMS (OF ORDER 2)

› prologue

› 2CPDS - a brief overview of results

› 2NW: a different way of understanding CPDS behaviours

› Nestify

› Spec languages: MSO, PDL, - UNDECIDABALITY

› under-approximations - tree interpretations - decidability
\[ \phi := a(x) \mid x < y \mid x \sim_1 y \mid x \sim_2 y \mid \phi \lor \phi \mid \neg \phi \mid \exists x \phi \mid \exists X \phi \mid x \in X \]

**EXAMPLE: CONSECUTIVE POPS LINKED TO THE SAME PUSH**

\[ \phi_{\rightarrow}(x, y) = \exists z \left( z \sim_1 x \land z \sim_1 y \land \neg \exists z' \left( z \sim_1 z' \land x < z' < y \right) \right) \]

**EXAMPLES:**

- set of all order-2 nested words
- language of an order-2 CPDS
- ...
SPECIFICATION FORMALISMS

MONADIC SECOND-ORDER LOGIC

\[ \phi := a(x) \mid x < y \mid x \sim_1 y \mid x \sim_2 y \mid \phi \lor \phi \mid \neg \phi \mid \exists x \phi \mid \exists X \phi \mid x \in X \]

SATISFIABILITY PROBLEM

input: MSO sentence \( \phi \)
question: is there a order-2 NW that satisfies \( \phi \)?

MODEL CHECKING PROBLEM

input: MSO sentence \( \phi \), order-2 CPDS \( H \)
question: do all order-2 NW from L(\( H \)) satisfy \( \phi \)?
SPECIFICATION FORMALISMS

MONADIC SECOND-ORDER LOGIC - UNDECIDABILITY

NOTICE: ORDER-2 STACK IS BOUNDED, NO COLLAPSE
PROPOSITIONAL DYNAMIC LOGIC WITH LOOP AND CONVERSE

\[ \varphi ::= a \mid \varphi \lor \varphi \mid \neg \varphi \mid \langle \pi \rangle \varphi \mid \text{Loop}(\pi) \]

\[ \pi ::= \{ \varphi \}\? \mid \rightarrow \mid \leftarrow \mid \rightarrow^1 \mid \rightarrow^1 \mid \rightarrow^2 \mid \rightarrow^2 \mid \pi \cdot \pi \mid \pi + \pi \mid \pi^* \]

\[ G, i \models \langle \pi \rangle \varphi \quad \text{if} \quad G, i, j \models \pi \quad \text{and} \quad G, j \models \varphi \quad \text{for some} \quad j \]

\[ G, i \models \text{Loop}(\pi) \quad \text{if} \quad G, i, i \models \pi \]

\[ G, i, j \models \{ \varphi \}? \quad \text{if} \quad i = j \quad \text{and} \quad G, i \models \varphi \]

\[ G, i, j \models \rightarrow \quad \text{if} \quad i \rightarrow j \quad \text{in the graph} \quad G \]

\[ G, i, j \models \rightarrow^1 \quad \text{if} \quad i \rightarrow^1 j \quad \text{in the graph} \quad G \]

\[ G, i, j \models \rightarrow^2 \quad \text{if} \quad j \rightarrow^2 i \quad \text{in the graph} \quad G. \]
PROPOSITIONAL DYNAMIC LOGIC WITH LOOP AND CONVERSE

- Why LCPDL?
- MSO powerful, but high complexity
- LTL less expressive, but low complexity
- LCPCL offers the best of both worlds (expressiveness, complexity)
PROPOSITIONAL DYNAMIC LOGIC WITH LOOP AND CONVERSE

EXAMPLE: PATH THAT GOES TO THE PUSH WHICH PUSHED THE CURRENT TOPMOST ELEMENT

\[(\{\uparrow_2 \lor ? \lor \text{nop}\}? \cdot \leftarrow + \leftarrow^1 + \leftarrow^2 \}^* \cdot \{\uparrow_1\}?)\]
PROPOSITIONAL DYNAMIC LOGIC WITH LOOP AND CONVERSE

EXAMPLE: PATH THAT GOES TO THE PUSH WHICH PUSHED THE CURRENT TOPMOST ELEMENT

\[
\{(\uparrow^2 \lor ? \lor \text{nop})? \cdot \leftarrow + (\uparrow^1 \cdot \leftarrow + \uparrow^2 \cdot \neg \text{Loop}(\leftarrow^+ \cdot \bowtie^2 \cdot \bowtie^2))? \cdot \leftarrow \}^* \cdot \{\uparrow_1\}?
\]
**LCPDL Sentence**

Boolean combinations of the form $E\varphi$

**Examples:**

- set of all order-2 nested words
- language of an order-2 CPDS
- …
**SPECIFICATION FORMALISMS**

**PROPOSITIONAL DYNAMIC LOGIC WITH LOOP AND CONVERSE**

### LCPDL SENTENCE

Boolean combinations of the form $E\varphi$

### SATISFIABILITY PROBLEM

**Input:** LCPDL sentence $\varphi$

**Question:** Is there a order-2 NW that satisfies $\varphi$?

### MODEL CHECKING PROBLEM

**Input:** LCPDL sentence $\varphi$, order-2 CPDS $H$

**Question:** Do all order-2 NW from $L(H)$ satisfy $\varphi$?
SPECIFICATION FORMALISMS

LCPDL UNDECIDABILITY

\[ \langle \leftrightarrow \rangle \varphi := \text{Loop}(\langle \{ \text{ispop}_1 \} \rangle ? \rightarrow )^+ \cdot \rightarrow \cdot \langle \{ \text{ispop}_2 \} \rangle ? \rightarrow \cdot \rightarrow \cdot (\rightarrow \cdot \langle \{ \text{ispop}_1 \} \rangle ?^+ \cdot \{ \varphi \} ? \cdot \text{\neg}^1 \cdot \text{\neg}^1) \]
OUTLINE

HIGHER ORDER PUSHDOWN SYSTEMS (OF ORDER 2)

- prologue
- 2CPDS - a brief overview of results
- 2NW: a different way of understanding CPDS behaviours
- Nestify
- Spec languages: MSO, PDL, - UNDECIDABALITY
- Eliminating collapse
- under-approximations - tree interpretations - decidability
Identifying valid encodings with overloading notations, we denote the translation of an node is labelled and path formulas connect a pair of representative positions. Checking whether the current LCPDL collapse will be in the language of 2-CPDS only on the nodes of interest. Further, in path formulas, moving right (1) would mean going to the next representative position on the right. Similarly for a node is labelled 2, would now amount to checking whether the current node is labelled by 2 without collapse. The other modalities remain unchanged. This translation can be done in 2-NW nodes.

\[
\begin{align*}
\bar{a} & \equiv a \lor \bar{a} \\
\varphi_1 \lor \varphi_2 & \equiv \varphi_1 \lor \varphi_2 \\
\neg \varphi & \equiv \neg \varphi \\
\langle \pi \rangle \varphi & \equiv \langle \pi \rangle \varphi \\
\text{Loop}(\pi) & \equiv \text{Loop}(\bar{\pi})
\end{align*}
\]
OUTLINE

HIGHER ORDER PUSHDOWN SYSTEMS (OF ORDER 2)

- prologue
- 2CPDS - a brief overview of results
- 2NW: a different way of understanding CPDS behaviours
- Nestify
- Spec languages: MSO, PDL, - UNDECIDABALITY
- under-approximations - tree interpretations - decidability
since the behaviors are graphs, we can use graph-based under-approximations. for example, tree-width.

decidability follows if the graph relations can be interpreted in a tree, and if the class is MSO definable. (Courcelle, also ParlatoMadhu, AGastin)

in our case they are even LCPDL definable (non-trivial, uses the characterisation)

can work on the tree domain and show that

- MSO SAT/MC is decidable
- PDL SAT/MC is ExpTime Complete
AN EXAMPLE: BOUNDED POP

- A pushed element can be popped at most $k$ times
- does not bound the size of the stacks in either level
- definable in LCPDL
- has bounded split-width (implies bounded tree-width)
CONCLUSION

HIGHER ORDER PUSHDOWN SYSTEMS (OF ORDER 2)

- prologue
- 2CPDS - a brief overview of results
- 2NW: a different way of understanding CPDS behaviours
- Nestify
- Spec languages: MSO, PDL, - UNDECIDABILITY
- Eliminating collapse
- under-approximations - tree interpretations - decidability
CONCLUSIONS AND PERSPECTIVES

- Yet another way of understanding HOPDS
- The power of data-flow edges. How to harness that?
- under-approximations
- applications
- lifting to higher orders
QUESTIONS?

THANK YOU

aiswarya @ cmi.ac.in