Priority Downward Closures

Theorietag ’23
appeared at CONCUR ’23

Ashwani Anand
Georg Zetzsche
Will the coffee machine go to a bad state?
[Haines] The system can be oversapproximated by a simpler system.
[Haines] The system can be oversimplified by a simpler system.
If a bad state is not reachable in new system it can not be reached in original system.
Haines] The system can be oversimplified by a simpler system.
If a bad state is not reachable in new system it cannot be reached in original system.
Can this system be overapproximated?
Can this system be overapproximated?

[This talk] Yes! For pushdown machines.
Can this system be overapproximated?

[This talk] Yes! For pushdown machines.
Precap

Block Order

Simple Machines

Pushdown Machines
Precap

Block Order

Simple Machines

Pushdown Machines
Subword Order
Subword Order
Subword Order

b b b b b b

r b b
Subword Order
Subword Order
Subword Order
Subword Order
Subword Order
Subword Order
Block Order
Block Order

0₆
1₄
0₆
1₆
0₆
0ₓ
Block Order
Block Order
Block Order
Block Order
Block Order
Block Order
Block Order

1r

1b

0b

0b

0b

0b

0b

0b
Block Order
Block Order

Diagram showing different notations and symbols related to block order.
Block Order
Block Order
Block Order
Block Order

Diagram showing bar graphs with varying heights.
Block Order
Block Order
Block Order

\[ \begin{aligned}
& 0^b \\
& 2^b \\
& 0^b \\
& 1^r \\
& 0^b \\
& 1^b \\
& 0^b \\
& 0^b \\
& \end{aligned} \]
Block Order
Block Order

Diagram showing a sequence of elements labeled 0b, 2b, 1r, 0b, 1b, 0b, 0b, 0b, 2r, 0b, 1r, 0b.
Block Order

- Split word into blocks.
Block Order

- Split word into blocks.
- Monotonic mapping of recursively embeddable blocks.
Block Order

- Split word into blocks.
- Monotonic mapping of recursively embeddable blocks.
- Highest priority letters occur in appropriate positions.
Block Order

- Split word into blocks.
- Monotonic mapping of recursively embeddable blocks.
- Highest priority letters occur in appropriate positions.

[In this talk] One letter per priority.
Block Order

Considers priority over letters
Block Order

Considers priority over letters

Refines Prioritised Superseding Order [Haase et al., 2014] and Subword Order
Block Downward Closures
Block Downward Closures

Sees only a block smaller queue of original queue.
Block Downward Closures

Sees only a block smaller queue of original queue.
Overapproximate to consider all block smaller behaviors shown by original machine.
Block Downward Closures

Downward closure of the machine.

Sees only a block smaller queue of original queue.

Overapproximate to consider all block smaller behaviors shown by original machine.
Block Downward Closures

Downward closure of the machine.

$$\forall \ell \in \mathcal{M}, \omega \in \mathcal{V} :$$

Sees only a block smaller queue of original queue.

Overapproximate to consider all block smaller behaviors shown by original machine.
Block Downward Closures

Downward closure of the machine:

$L \downarrow = \{ w \mid \text{vel, } w \in \mathcal{L} \}$

Always accepted by finite state machine.

Sees only a block smaller queue of original queue.

Overapproximate to consider all block smaller behaviors shown by original machine.
Block Downward Closures

Downward closure of the machine.

$L_{LL} = \{ w \mid \text{vel}, w \in \sum^* \}$

Always accepted by finite state machine.

Might not be computable!
Block Downward Closures

Downward closure of the machine.

$L = \{ w \mid \text{vel}, \text{vel} \}\$

Always accepted by finite state machine.

Might not be computable!

[[This talk] Construction of such machine for pushdown machines.]
Coffee machine at a concert
Block Downward Closures

Coffee machine at a concert

Network of systems with congestion controlled communication channels
Overview

Block Order

- Considers priorities
- Refines subword order and PSO

Simple Machines

Pushdown Machines
Overview

Block Order
- Considers priorities
- Refines subword order and PSO

Simple Machines

Pushdown Machines
Finite State Machines

Finite state machine

accepting a language $L$
Finite State Machines

Finite state machine
accepting a language $L$

A transducer
takes an input word, outputs another word.
Finite State Machines

Finite state machine
accepting a language $L$

A transducer
takes an input word, outputs another word.

Another finite state machine
accepting block downward closure $L^D$
Finite State Machines

A transducer

takes an input word, outputs another word.

The transducer is computable in polytime.

Downward closures can be computed in polytime.
One Counter Machines

FSA with a counter with zero tests
One Counter Machines

FSA with a counter with zero tests

FSA with a counter without zero tests
One Counter Machines

FSA with a counter with zero tests

FSA with a counter without zero tests

FSA recognizing block downward closure.
One Counter Machines

FSA with a counter with zero tests

FSA with a counter without zero tests

FSA recognizing block downward closure.

Keep track of the counter for a fixed polynomial bound.
One Counter Machines

FSA with a counter with zero tests

FSA with a counter without zero tests

FSA recognizing block downward closure.

- Keep track of the counter for a fixed polynomial bound.

- If bound is exceeded, there is a cycle which increases counter and one that decreases.
One Counter Machines

FSA with a counter with zero tests → FSA with a counter without zero tests → FSA recognizing block downward closure.

Block downward closure for an OCA language can be computed in polytime.
Overview

Block Order

- Considers priorities
- Refines subword order and PSO

Simple Machines

- Downward closures computable in polytime

Pushdown Machines
Overview

Block Order
- Considers priorities
- Refines subword order and PSO

Simple Machines
- Downward closures computable in polytime

Pushdown Machines
Context Free Languages

\[ S \rightarrow ASB | \varepsilon \]
\[ A \rightarrow 010 \]
\[ B \rightarrow 212 \]
Context Free Languages

\[ S \rightarrow ASB \mid \varepsilon \]

\[ A \rightarrow 010 \]

\[ B \rightarrow 212 \]
Context Free Languages

\[ S \rightarrow ASB \mid \varepsilon \]
\[ A \rightarrow 010 \]
\[ B \rightarrow 212 \]
Context Free Languages

$S \rightarrow ASB \mid \epsilon$

$A \rightarrow 010$

$B \rightarrow 212$
Context Free Languages

S → ASB | ε
A → 010
B → 212
Context Free Languages

\[ S \rightarrow ASB | \epsilon \]
\[ A \rightarrow 010 \]
\[ B \rightarrow 212 \]

\[ (010)^n (212)^n \]
Context Free Languages

\[ S \rightarrow AB \mid \epsilon \]
\[ A \rightarrow 010 \]
\[ B \rightarrow 212 \]
\[ (010)^n (212)^n \]

Derivation tree can have arbitrary depth.
Context Free Languages

\[ S \rightarrow ASB \mid \epsilon \]
\[ A \rightarrow 010 \]
\[ B \rightarrow 212 \]

\((010)^n (212)^n\)

Derivation tree can have arbitrary depth.

We try "bounding" the depth.
Context Free Languages

Context free grammar
Context Free Languages

Context free grammar
Context Free Languages

Context free grammar

Another grammar that
- has same downward closure
Context Free Languages

Context free grammar

Another grammar that
- has same downward closure
- any word can be generated by "bounded" depth derivation trees.
Context Free Languages

\[ S \rightarrow ASB | \epsilon \]

\[ A \rightarrow 010 \]

\[ B \rightarrow 212 \]

\[(010)^n (212)^n\]
Context Free Languages

\[ S \rightarrow ASB \mid \varepsilon \]
\[ A \rightarrow 010 \]
\[ B \rightarrow 212 \]

\[ (010)^n (212)^n \]

[Diagram: A tree with the root labeled \( S \) and other nodes labeled with productions and symbols like \( (010)^n \) and \( (212)^n \)]
Context Free Languages

\[ S \rightarrow ASB \mid \epsilon \]
\[ A \rightarrow 010 \]
\[ B \rightarrow 212 \]

\[(010)^n (212)^n\]

\(s\)

Compute block downward closures of these cycles.
Context Free Languages

$(010)^n \# (212)^n$
Context Free Languages

\[(010)^n \# (212)^n\]

\[(010)^n \downarrow\]
Context Free Languages

\[(010)^n \uparrow \# \ (212)^n\]

\[(010)^n \downarrow = L \uparrow 1\]

↓

Set of first blocks
Context Free Languages

\[(010)^n \# (212)^n\]

\[(010)^n \uparrow = \{ \uparrow 1 \cdot (M \uparrow 1)^* \}

Set of first blocks

Set of blocks surrounded by 1s
Context Free Languages

\[(010)^n \# \ (212)^n\]

\[(010)^n \Rightarrow 1 \cdot (M \ddownarrow 1)^* \cdot R \Uparrow\]

- Set of first blocks
- Set of blocks surrounded by 1s
- Set of last blocks
Context Free Languages

\[(010)^n \# (212)^n\]

\[(010)^n \downarrow = L \downarrow 1 \cdot (M \downarrow 1)^* \cdot R \downarrow\]

- Set of first blocks
- Set of blocks surrounded by 1s
- Set of last blocks

L, M, R have one less priority
Context Free Languages

\[(010)^n \# (212)^n\]

\[(010)^n \downarrow = L \downarrow 1 \cdot (M \downarrow 1)^* \cdot R \downarrow\]

- set of first blocks
- set of blocks surrounded by 1s
- set of last blocks

\[= \{ \epsilon, 031 \cdot (\{ \epsilon, 0, 00\} 1)^* \cdot \{ \epsilon, 03 \}\} \]
Context Free Languages

\[(010)^n \# (212)^n\]

\[(010)^n \downarrow = L \uparrow 2 \cdot (M \downarrow 2)^* \cdot R \downarrow\]

- Set of first blocks
- Set of blocks surrounded by 2s
- Set of last blocks

\[= \{ \varepsilon \} \cdot (\{ \varepsilon, 1 \} \downarrow 2)^* \cdot \{ \varepsilon \} \]
Context Free Languages

S → A S B I E
A → 010
B → 212

Another grammar that
• has same downward closure
• any word can be generated by "bounded" depth derivation trees.
Context Free Languages

\[ S \rightarrow ASB \mid \varepsilon \]
\[ A \rightarrow 010 \]
\[ B \rightarrow 212 \]

\[ S \rightarrow ASB \mid \varepsilon \]
\[ A \rightarrow 010 \]
\[ B \rightarrow 212 \]
Context Free Languages

S → ASB | ε | A* # B*
A → 010
B → 212

S → ASB | ε | A* # B*
A → 010
B → 212
\overline{A} → \{e,0\}^1 (\{e,0,00\}^* 1)^* \{e,0\}
\overline{B} → 2 \cdot (\{e,1\}^* 2)^*
Context Free Languages

\[
S \rightarrow ASB \mid \varepsilon \mid \bar{A}^* \# \bar{B}^*
\]

\[
A \rightarrow 010
\]

\[
B \rightarrow 212
\]

\[
\bar{A} \rightarrow \{ \varepsilon, 0 \} (\{ \varepsilon, 0, 0001 \}^* \varepsilon, 03^2
\]

\[
\bar{B} \rightarrow 2. (\{ \varepsilon, 13, 2 \}^*)
\]

\[
\# \rightarrow \varepsilon
\]
Context Free Languages

S → ASB | ε | A* # B*
A → 010
B → 212

Both grammars have same block downward closures.
Context Free Languages

\[
S \rightarrow ASB \mid \varepsilon \\
A \rightarrow 010 \\
B \rightarrow 212
\]

Exponential blowup.

\[
S \rightarrow ASB \mid \varepsilon \mid \overline{A}^* \# \overline{B}^* \\
A \rightarrow 010 \\
B \rightarrow 212 \\
\overline{A} \rightarrow \{\varepsilon,0\}^* \{\varepsilon,0,000,1\}^* \{\varepsilon,0\}^* \\
\overline{B} \rightarrow 2 \cdot \{\varepsilon,13,2\}^* \\
\# \rightarrow \varepsilon
\]
Context Free Languages

\[ S \rightarrow ASB \mid \epsilon \]
\[ A \rightarrow 010 \]
\[ B \rightarrow 212 \]

Exponential blowup.

\[ S \rightarrow ASB \mid \epsilon \mid \overline{A}^* \# \overline{B}^* \]
\[ A \rightarrow 010 \]
\[ B \rightarrow 212 \]
\[ \overline{A} \rightarrow \{e, 0\}^* \{e, 0, 000, 1\}^* \{e, 0\}^* \]
\[ \overline{B} \rightarrow 2 \cdot \{e, 13, 2\}^* \]
\[ \# \rightarrow \epsilon \]

Can be recognized by a finite state machine of exponential size.
Context Free Languages

Downward closure of a CFG can be recognized by a FSM of doubly exponential size.
Context Free Languages

Downward closure of a CFG can be recognized by a FSM of doubly exponential size.

Exponential lower bound is inherited from subword order.
Summary

Block Order

- Considers priorities
- Refines subword order and PSO

Simple Machines

- Downward closures computable in polytime

Pushdown Machines

- 2-EXP upper bound
- Exp lower bound