Priority Downward Closures

CONCUR’23

Ashwani Anand
Georg Zetzsche
Will the coffee machine go to a bad state?
[Haines] The system can be oversimplified 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 cannot 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 can not 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 oversimplified?

[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
Subword Order
Subword Order
Subword Order
Subword Order
Subword Order

![Diagram showing subword order with characters b, τ, b, b, b, and τ in various positions.]

?
Block Order
Block Order
Block Order
Block Order
Block Order
Block Order
Block Order
Block Order
Block Order
Block Order
Block Order
Block Order
Block Order
Block Order
Block Order

Diagram showing a series of bars and arrows indicating a block order.
Block Order
Block Order

Diagram showing various blocks and their relationships with arrows indicating connections.
Block Order

- Split word into blocks.
Block Order

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

Monotonic mapping of recursively embeddable blocks.

Highest priority letters occur in appropriate positions.
Block Order

[In this talk] One letter per priority.

- Split word into blocks.
- Monotonic mapping of recursively embeddable blocks.
- Highest priority letters occur in appropriate positions.
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:
\[ L' = \{ w | \text{w is a prefix of a } \in L \} \]

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:

$L_D = \{ w \mid \forall w' \in L, w \leq w' \}$

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.

$L^1 = \{w \mid v \in L, w \in B^*v\}$

Always accepted by finite state machine.

Might not be computable!
Block Downward Closures

Downward closure of the machine.

$L_{M} = \{ w | v \in L, w \in B v \}$

Always accepted by finite state machine.

Might not be computable!

[This talk] Construction of such machine for pushdown machines.
Block Downward Closures

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 $L_1$.
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 \( L \upharpoonright \upharpoonright \)

Construct a transducer that outputs block smaller words for a given 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 $L_\downarrow$

Construct a transducer that outputs block smaller words for a given word.
Transducer for Block Order
Transducer for Block Order

[Diagram]

[Idea] If i is skipped then skip all j<i, until an i is output again.
Transducer for Block Order

0, ε, 0
1, 2
2, ε

1
0, ε
1, ε, 1
2, ε

2
0, ε
1, ε, 1
2, ε, 2

0 2 0 1 0 1 0 0 0
Transducer for Block Order
Transducer for Block Order
Transducer for Block Order
Transducer for Block Order

Diagram showing a transition graph with states labeled 0, 1, and 2, and transitions labeled with symbols like $\epsilon$, 1, 2, and 0.
Transducer for Block Order
Transducer for Block Order
Transducer for Block Order
Transducer for Block Order

Diagram showing transitions and states with labels like 0, ε, 0², 1, 1, 2, 2, 1, ε, 2, ε, 1, ε, 1², 2, ε, 2².
Transducer for Block Order

Diagram:

States:
- 1
- 2

Transitions:
- 0, ε, 0^2
- 1, 1
- 2, 2
- 1, ε
- 2, ε
- 0, ε
- 1, ε
- 2, ε
- 0^3

Output:
- 0, 2, 0, 1, 0, 1, 0, 0, 0

Arrow:
- ↑

Notes:
- 2, 0
Transducer for Block Order

Diagram showing states and transitions with input symbols 0, 1, and 2.
Transducer for Block Order

Diagram of a transducer with states 1 and 2, showing transitions for inputs 0, ε, 1, 2, and outputs 0, 1, 2, ε.

Bar chart representing the output sequence for input sequence 0, 1, 0, 1, 0, 1, 0, 0, 0.
Transducer for Block Order

Diagram:

- States: 0, 1, 2
- Transitions:
  - 0: ε, 0
  - 1: 1, 1
  - 2: 2, ε

Graph:

- Edges:
  - 0 → 1: 1, 1
  - 0 → 2: 2, ε
  - 1 → 0: ε
  - 1 → 2: 1, ε
  - 2 → 0: 1, ε
  - 2 → 1: 2, ε

- Initial state: 0
- Final states: 1, 2

Bar chart:

- Bars for each state:
  - 0: 2
  - 1: 1
  - 2: 1

- Transition arrows:
  - From 0 to 1: 1, 1
  - From 0 to 2: 2, ε
  - From 1 to 0: ε
  - From 1 to 2: 1, ε
  - From 2 to 0: 1, ε
  - From 2 to 1: 2, ε

- Arrow pointing to 1
Transducer for Block Order

Diagram showing states and transitions:
- States: 1, 2
- Transitions:
  - 0, ε, ε, 0^3
  - 1, 1
  - 2, 2
- Input symbols: 0, 1, 2, ε

Graphical representation of transitions and states.

Bar chart showing output values:
- Outputs: 0, 2, 0, 1, 0, 1, 0, 0
Transducer for Block Order
Transducer for Block Order
Transducer for Block Order
Transducer for Block Order

Diagram:

- States: 1, 2
- Transitions:
  - 0, ε, 0³
  - 1, ε, 1
  - 2, ε, 1

- Initial State: 1
- Final States:
  - 1
  - 2

- Bar Graph:
  - Values: 0, 2, 0, 1, 0, 1, 0, 0, 0
  - Arrows: ↑ from 2 to 0
Transducer for Block Order

Diagram:

- States: 0, 1, 2
- Transitions:
  - 0 → 1: ε, 1
  - 1 → 0: ε, 1
  - 1 → 2: ε, ε
  - 2 → 0: ε, ε
  - 2 → 2: ε, ε

Graph:

- Nodes: 0, 1, 2
- Edges:
  - 0 → 1
  - 1 → 0
  - 1 → 2
  - 2 → 0
  - 2 → 2
- Bars:
  - 0: 2
  - 1: 0, 1
  - 2: 0, 0

Sequence:

- 0, 2, 0, 1, 0, 1, 0, 0, 0

Arrows:

- 0 → 1
- 1 → 0
- 1 → 2
- 2 → 0
- 2 → 2

Transducer for Block Order
Transducer for Block Order
Transducer for Block Order

```
0, ε, 0
1, 1, 1
2, 2, 2
1, ε
1, ε
1, ε
2, ε
0, ε
```

```
0 2 0 1 0 1 0 0 0
```

```
0
↑
```

```
0
↑
```
Transducer for Block Order
Transducer for Block Order
Transducer for Block Order
Transducer for Block Order
Transducer for Block Order

The transducer is computable in polytime.
The transducer is computable in polytime.

Downword closures for FSM can be computed in polytime.
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 → ASB | ε
A → 010
B → 212
Context Free Languages

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

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

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

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

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

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

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

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

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 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

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

\((010)^* (212)^*\)
Context Free Languages

\[ S \rightarrow A S B \mid \varepsilon \]

\[ A \rightarrow 010 \]

\[ B \rightarrow 212 \]

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

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

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

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

$(010)^n \# (212)^n$

$(010)^n \downarrow = L \uparrow 1$

\[\downarrow\]

Set of first blocks
Context Free Languages

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

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

\[
\begin{align*}
\text{Set of} & \quad \text{Set of} \\
\text{first blocks} & \quad \text{blocks} \\
& \quad \text{surrounded} \\
& \quad \text{by 1s}
\end{align*}
\]
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
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

\[= \{\varepsilon, 031 \cdot (\varepsilon, 0, 0031)^* \cdot \varepsilon, 03\} \]
Context Free Languages

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

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

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

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

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

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

\[
\begin{align*}
S & \rightarrow ASB | \epsilon \\
A & \rightarrow 010 \\
B & \rightarrow 212 \\
S & \rightarrow ASB | \epsilon \\
A & \rightarrow 010 \\
B & \rightarrow 212
\end{align*}
\]
Context Free Languages

\[ S \rightarrow AB \mid \varepsilon \mid A^* \# B^* \]

\[ A \rightarrow 010 \]

\[ B \rightarrow 212 \]

\[ \overline{A} \rightarrow \{\varepsilon, 03\}^* (\{\varepsilon, 0, 003, 1\})^* \varepsilon, 03^* \]

\[ \overline{B} \rightarrow 2 (\{\varepsilon, 13, 2\})^* \]
Context Free Languages

\[ S \rightarrow ASB | \varepsilon | A^* \# \bar{B}^* \]
\[ A \rightarrow 010 \]
\[ B \rightarrow 212 \]

\[ \bar{A} \rightarrow \{ \varepsilon, 0, 3 \} \cdot (\{ \varepsilon, 0, 1 \})^* \cdot \varepsilon, 0, 3 \]
\[ \bar{B} \rightarrow 2 \cdot (\{ \varepsilon, 1, 3 \}^* \# \varepsilon \]

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

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

\[
A \rightarrow 010
\]

\[
B \rightarrow 212
\]

\[
\overline{A} \rightarrow \{\varepsilon, 031 (\varepsilon, 0, 003.1)^* \varepsilon, 03 \}
\]

\[
\overline{B} \rightarrow 2 (\varepsilon, 13.2)^*
\]

\[
\# \rightarrow \varepsilon
\]

Both grammars have same block downward closures.
Context Free Languages

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

Exponential blowup.

\[ S \rightarrow ASB | \epsilon | A^* \# B^* \]
\[ A \rightarrow 010 \]
\[ B \rightarrow 212 \]
\[ \overline{A} \rightarrow \{\epsilon, 03\} \cup \{\epsilon, 0, 003 \# \epsilon, 03\} \]
\[ \overline{B} \rightarrow 2 \cdot (\{\epsilon, 13 \# 2\})^* \]
\[ \# \rightarrow \epsilon \]
Context Free Languages

\[ S \rightarrow ASB \mid \epsilon \mid \overline{A}^* \# \overline{B}^* \]

\[ A \rightarrow 010 \]

\[ B \rightarrow 212 \]

Exponential blowup.

\[ \overline{A} \rightarrow \{\epsilon, 0, 1\} \{\epsilon, 0, 00, 0, 1\}^* \{\epsilon, 0, \epsilon\} \]

\[ \overline{B} \rightarrow 2 \{\epsilon, 1, 3 \cdot 2\}^* \]

\[ \# \rightarrow \epsilon \]

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