Downward closures and complexity

Georg Zetzsche\textsuperscript{1}

LSV, CNRS & ENS Cachan
Université Paris-Saclay

Higher-Order Model Checking
Shonan Meeting, March 14-17, 2016

\textsuperscript{1}Supported by a fellowship within the Postdoc-Program of the German Academic Exchange Service (DAAD).
System Observer

Downward Closures

\[ u \vdash v : u \text{ is a subsequence of } v \]

Observer sees precisely

Georg Zetzsche (LSV Cachan)
System Observer
LOSSY CHANNEL

Downward Closures

\( u \) is a subsequence of \( v \):

Observer sees precisely

Georg Zetzsche (LSV Cachan)
System Observer
LOSSY
CHANNEL

Downward Closures

$u \ddot{\epsilon} v$: $u$ is a subsequence of $v$

Observer sees precisely $L = u$

Georg Zetzsche (LSV Cachan)

Shonan HOMC 2 / 16
**Downward Closures**

- $u \preceq v$: $u$ is a subsequence of $v$
- $L\downarrow = \{ u \in X^* \mid \exists v \in L : u \preceq v \}$
- Observer sees precisely $L\downarrow$
Downward Closures

Theorem (Higman/Haines)

For every language \( L \subseteq X^* \), \( L \downarrow \) is regular.

Applications

Given an automaton for \( L \downarrow \), many things are decidable:

- Inclusion of behavior under lossy observation (\( K \subseteq \bar{X} \))

  Ordinary inclusion almost always undecidable!

- Which actions occur arbitrarily often? (\( a \subseteq \bar{X} \))

- Can the system run arbitrarily long? (\( L \) infinite)

Safety verification of parametrized asynchronous shared-memory systems (La Torre, Muscholl, Walukiewicz, FSTTCS 2015)
Downward Closures

Theorem (Higman/Haines)

For every language $L \subseteq X^*$, $L\downarrow$ is regular.

Applications

Given an automaton for $L\downarrow$, many things are decidable:
Downward Closures

Theorem (Higman/Haines)

For every language $L \subseteq X^*$, $L\downarrow$ is regular.

Applications

Given an automaton for $L\downarrow$, many things are decidable:

- Inclusion of behavior under lossy observation ($K\downarrow \subseteq L\downarrow$)
  Ordinary inclusion almost always undecidable!
**Downward Closures**

**Theorem (Higman/Haines)**

*For every language \( L \subseteq X^* \), \( L \downarrow \) is regular.*

**Applications**

Given an automaton for \( L \downarrow \), many things are decidable:

- Inclusion of behavior under lossy observation \( (K \downarrow \subseteq L \downarrow) \)
  
  Ordinary inclusion almost always undecidable!

- Which actions occur arbitrarily often? \( (a^* \subseteq L \downarrow) \)
Downward Closures

Theorem (Higman/Haines)

For every language $L \subseteq X^*$, $L\downarrow$ is regular.

Applications

Given an automaton for $L\downarrow$, many things are decidable:

- Inclusion of behavior under lossy observation ($K\downarrow \subseteq L\downarrow$)
  Ordinary inclusion almost always undecidable!
- Which actions occur arbitrarily often? ($a^* \subseteq L\downarrow$)
- Can the system run arbitrarily long? ($L\downarrow$ infinite)
**Downward Closures**

**Theorem (Higman/Haines)**

For every language $L \subseteq X^*$, $L\downarrow$ is regular.

**Applications**

Given an automaton for $L\downarrow$, many things are decidable:

- Inclusion of behavior under lossy observation ($K\downarrow \subseteq L\downarrow$)
  Ordinary inclusion almost always undecidable!
- Which actions occur arbitrarily often? ($a^* \subseteq L\downarrow$)
- Can the system run arbitrarily long? ($L\downarrow$ infinite)
- Safety verification of parametrized asynchronous shared-memory systems (La Torre, Muscholl, Walukiewicz, FSTTCS 2015)
Downward Closures

Theorem (Higman/Haines)

For every language $L \subseteq X^*$, $L\downarrow$ is regular.

Applications

Given an automaton for $L\downarrow$, many things are decidable:

- Inclusion of behavior under lossy observation ($K\downarrow \subseteq L\downarrow$)
  Ordinary inclusion almost always undecidable!
- Which actions occur arbitrarily often? ($a^* \subseteq L\downarrow$)
- Can the system run arbitrarily long? ($L\downarrow$ infinite)
- Safety verification of parametrized asynchronous shared-memory systems (La Torre, Muscholl, Walukiewicz, FSTTCS 2015)

Problem

- Finite automaton for $L\downarrow$ exists for every $L$.
- How can we compute it?
A general approach

Example (Transducer)

\[ \begin{align*}
\varepsilon & | a, \varepsilon | b \\
q_0 & \quad \rightarrow \quad \varepsilon | \# \\
q_1 & \quad \rightarrow \quad \varepsilon | \# \\
q_2 & \quad \rightarrow \quad \varepsilon | a, \varepsilon | b \\
\varepsilon & | a, \varepsilon | b \\
a & | a, b | b \\
\varepsilon & | \varepsilon, b | \varepsilon \\
a & | \varepsilon, b | \varepsilon \\
\varepsilon & | \varepsilon, b | \varepsilon \\
a & | \varepsilon, b | \varepsilon \\
\end{align*} \]

Definition

Rational transduction: set of pairs given by a finite state transducer. For rational transduction \( \mathcal{T} \in \mathcal{X} \rightarrow \mathcal{Y} \) and language \( \mathcal{L} \in \mathcal{Y} \rightarrow \mathcal{L} \), let \( T \mathcal{L} = \{ (p, x, y, q) : p \in \mathcal{T} \} \).
A general approach

Example (Transducer)

\[ T(A) = \{(x, u\#v\#w) \mid u, v, w, x \in \{a, b\}^*, \ v \leq x\} \]
A general approach

Example (Transducer)

\[
\begin{align*}
\varepsilon &\mid a, \varepsilon \mid b \\
\varepsilon &\mid \# \\
q_0 &\rightarrow q_1 \\
a &\mid \varepsilon, b &\mid \varepsilon \\
q_1 &\rightarrow q_2 \\
\varepsilon &\mid \# \\
a &\mid \varepsilon, b &\mid \varepsilon
\end{align*}
\]

\[T(A) = \{(x, u\#v\#w) \mid u, v, w, x \in \{a, b\}^*, \ v \leq x\}\]

Definition

- **Rational transduction**: set of pairs given by a finite state transducer.
- For rational transduction \( T \subseteq X^* \times Y^* \) and language \( L \subseteq Y^* \), let

\[
TL = \{ y \in Y^* \mid \exists x \in L : (x, y) \in T \}
\]
Definition

$C$ is a full trio if $LR \in C$ for each $L \in C$ and rational transduction $R$. 

Theorem (Z., ICALP 2015)

If $C$ is a full trio, then downward closures are computable for $C$ if and only if the simultaneous unboundedness problem is decidable:

Given a language $L \in \mathcal{C}$, question is:

Is $a_{1} \cdot \ldots \cdot a_{n}$ included in $L$?

Equivalently, we check whether it is true that:

for each $k \geq 0$, there are $x_{1}, \ldots, x_{n} \geq k$ with $a_{x_{1}} \cdot \ldots \cdot a_{x_{n}} \in L$. 

Georg Zetzsche (LSV Cachan)
**Definition**

$C$ is a **full trio** if $LR \in C$ for each $L \in C$ and rational transduction $R$.

---

**Theorem (Z., ICALP 2015)**

If $C$ is a full trio, then downward closures are computable for $C$ if and only if the **simultaneous unboundedness problem** is decidable:

*Given* A language $L \subseteq a_1^* \cdots a_n^*$ in $C$

*Question* Is $a_1^* \cdots a_n^*$ included in $L\downarrow$?
Definition

$C$ is a **full trio** if $LR \in C$ for each $L \in C$ and rational transduction $R$.

Theorem (Z., ICALP 2015)

*If $C$ is a full trio, then downward closures are computable for $C$ if and only if the *simultaneous unboundedness problem* is decidable:*

**Given** A language $L \subseteq a_1^* \cdots a_n^*$ in $C$

**Question** Is $a_1^* \cdots a_n^*$ included in $L\downarrow$?

Equivalently, we check whether it is true that:

for each $k \geq 0$, there are $x_1, \ldots, x_n \geq k$ with $a_1^{x_1} \cdots a_n^{x_n} \in L$
Theorem (Jullien 1969, Abdulla et. al. 2004)

Every language $L$ can be written as a finite union of ideals:

$$Y_0^* \{x_1, \varepsilon\} Y_1^* \cdots \{x_n, \varepsilon\} Y_n^*,$$

where $x_1, \ldots, x_n$ are letters and $Y_0, \ldots, Y_n$ are alphabets.
Theorem (Jullien 1969, Abdulla et. al. 2004)

Every language $L$ can be written as a finite union of ideals:

$$Y_0^*\{x_1, \varepsilon\} Y_1^* \cdots \{x_n, \varepsilon\} Y_n^*,$$

where $x_1, \ldots, x_n$ are letters and $Y_0, \ldots, Y_n$ are alphabets.

Ideal decompositions: currently also studied by Lazić, Leroux, Schmitz
Theorem (Jullien 1969, Abdulla et. al. 2004)

Every language $L$ can be written as a finite union of ideals:

$$Y_0^*\{x_1, \varepsilon\} Y_1^* \cdots \{x_n, \varepsilon\} Y_n^*,$$

where $x_1, \ldots, x_n$ are letters and $Y_0, \ldots, Y_n$ are alphabets.

Ideal decompositions: currently also studied by Lazić, Leroux, Schmitz

Algorithm

Suppose $L \subseteq X^*$ is given.
Enumerate simple regular languages $R$.
Decide whether $L \downarrow = R$:
Theorem (Jullien 1969, Abdulla et. al. 2004)

Every language $L\downarrow$ can be written as a finite union of ideals:

$$Y_0^*\{x_1, \varepsilon\} Y_1^* \cdots \{x_n, \varepsilon\} Y_n^*,$$

where $x_1, \ldots, x_n$ are letters and $Y_0, \ldots, Y_n$ are alphabets.

Ideal decompositions: currently also studied by Lazić, Leroux, Schmitz

Algorithm

Suppose $L \subseteq X^*$ is given.
Enumerate simple regular languages $R$.
Decide whether $L\downarrow = R$:

- $L\downarrow \subseteq R$ iff $L\downarrow \cap (X^* \setminus R) = \emptyset \iff$ emptiness.
Theorem (Jullien 1969, Abdulla et. al. 2004)

Every language $L$ can be written as a finite union of ideals:

$$Y_0^*\{x_1, \varepsilon\} Y_1^* \cdots \{x_n, \varepsilon\} Y_n^*,$$

where $x_1, \ldots, x_n$ are letters and $Y_0, \ldots, Y_n$ are alphabets.

Ideal decompositions: currently also studied by Lazić, Leroux, Schmitz

Algorithm

Suppose $L \subseteq X^*$ is given.
Enumerate simple regular languages $R$.
Decide whether $L = R$:

- $L \subseteq R$ iff $L \cap (X^* \setminus R) = \emptyset \iff$ emptiness.

Observation

$L \downarrow$ is in $C$:

$$(x, \varepsilon)$$

$$(x, x)$$
Theorem (Jullien 1969, Abdulla et. al. 2004)

Every language $L$ can be written as a finite union of ideals:

$$Y_0^*\{x_1, \varepsilon\} Y_1^* \cdots \{x_n, \varepsilon\} Y_n^*,$$

where $x_1, \ldots, x_n$ are letters and $Y_0, \ldots, Y_n$ are alphabets.

Ideal decompositions: currently also studied by Lazić, Leroux, Schmitz

Algorithm

Suppose $L \subseteq X^*$ is given.

Enumerate simple regular languages $R$.

Decide whether $L \downarrow = R$:

- $L \downarrow \subseteq R$ iff $L \downarrow \cap (X^* \setminus R) = \emptyset \rightsquigarrow$ emptiness.
- $R \subseteq L \downarrow \rightsquigarrow Y_0^*\{x_1, \varepsilon\} Y_1^* \cdots \{x_n, \varepsilon\} Y_n^* \subseteq L \downarrow$

Observation

$L \downarrow$ is in $C$:

- $(x, \varepsilon)$
- $(x, x)$
Observation

- It suffices to check whether \( Y_0^* \{ x_1, \varepsilon \} Y_1^* \cdots \{ x_n, \varepsilon \} Y_n^* \subseteq L\downarrow. \)
Observation

- It suffices to check whether $Y_0^* \{x_1, \varepsilon\} Y_1^* \cdots \{x_n, \varepsilon\} Y_n^* \subseteq L\downarrow$.
- $L\downarrow$ includes $\{a, b, c\}^*$ if and only if it contains $(abc)^*$.
Observation

- It suffices to check whether \( Y_0^* \{ x_1, \varepsilon \} Y_1^* \cdots \{ x_n, \varepsilon \} Y_n^* \subseteq L\downarrow \).
- \( L\downarrow \) includes \( \{ a, b, c \}^* \) if and only if it contains \( (abc)^* \).

\[ abc \quad abc \quad abc \quad abc \quad abc \quad abc \]
Observation

- It suffices to check whether $Y_0^*\{x_1, \varepsilon\} Y_1^* \cdots \{x_n, \varepsilon\} Y_n^* \subseteq L\downarrow$.
- $L\downarrow$ includes $\{a, b, c\}^*$ if and only if it contains $(abc)^*$.

```
abc abc abc abc abc abc
bacca
```
Observation

- It suffices to check whether $Y_0^* \{x_1, \varepsilon\} Y_1^* \cdots \{x_n, \varepsilon\} Y_n^* \subseteq L\downarrow$.
- $L\downarrow$ includes $\{a, b, c\}^*$ if and only if it contains $(abc)^*$.

```
abc abc abc abc abc
bacca
```
Observation

- It suffices to check whether $Y_0^* \{ x_1, \varepsilon \} Y_1^* \cdots \{ x_n, \varepsilon \} Y_n^* \subseteq L\downarrow$.
- $L\downarrow$ includes $\{ a, b, c \}^*$ if and only if it contains $(abc)^*$.

\[ abc \; abc \; abc \; abc \; abc \; abc \]
\[ bacca \]

Transduction $T$

\[ \begin{array}{c}
q_0 \xrightarrow{y_0|a_0} \quad x_1 \xrightarrow{y_1|a_1} \quad x_2 \xrightarrow{y_2|a_2} \cdots \quad x_n \xrightarrow{y_n|a_n} \quad q_n
\end{array} \]

$y_i$: word containing each letter of $Y_i$ once.
Observation

- It suffices to check whether \( Y_0^* \{ x_1, \varepsilon \} Y_1^* \cdots \{ x_n, \varepsilon \} Y_n^* \subseteq L\downarrow. \)
- \( L\downarrow \) includes \( \{a, b, c\}^* \) if and only if it contains \((abc)^*\).

\[
abc \ abc \ abc \ abc \ abc \ abc
\]

\[
bacca
\]

Transduction \( T \)

\[
y_0 | a_0 \quad \xrightarrow{x_1 | \varepsilon} \quad y_1 | a_1 \quad \xrightarrow{x_2 | \varepsilon} \cdots \quad y_n | a_n \quad \xrightarrow{x_n | \varepsilon} \quad q_n
\]

\( y_i \): word containing each letter of \( Y_i \) once. Then:

\[
T(L\downarrow)\downarrow = a_0^* \cdots a_n^* \iff Y_0^* \{ x_1, \varepsilon \} Y_1^* \cdots \{ x_n, \varepsilon \} Y_n^* \subseteq L\downarrow
\]
- New algorithm for every known computable case
- New algorithm for every known computable case
- Additional language classes:
- New algorithm for every known computable case
- Additional language classes:

**Corollary (Z., ICALP 2015)**

*Downward closures are computable for order-2 pushdown automata.*
New algorithm for every known computable case

Additional language classes:

**Corollary (Z., ICALP 2015)**

*Downward closures are computable for order-2 pushdown automata.*

**Theorem (Hague, Kochems, Ong, POPL 2016)**

*Downward closures are computable for higher-order pushdown automata.*
New algorithm for every known computable case
Additional language classes:

**Corollary (Z., ICALP 2015)**

*Downward closures are computable for order-2 pushdown automata.*

**Theorem (Hague, Kochems, Ong, POPL 2016)**

*Downward closures are computable for higher-order pushdown automata.*

Igor’s talk: higher-order recursion schemes
Higher-Order Pushdown Automata

Let $\Gamma$ be a stack alphabet. 
$S_\Gamma^n$ is the set of order-$n$ stacks:

$$S_\Gamma^0 = \Gamma \quad S_\Gamma^{k+1} = \{ [s_1 \cdots s_m]_{k+1} \mid s_1, \ldots, s_m \in S_\Gamma^k \}.$$
Higher-Order Pushdown Automata

Let $\Gamma$ be a stack alphabet. $S_\Gamma^n$ is the set of order-$n$ stacks:

$$S_0^\Gamma = \Gamma \quad S_{k+1}^\Gamma = \{[s_1 \cdots s_m]_{k+1} \mid s_1, \ldots, s_m \in S_k^\Gamma\}.$$

**Operations on order-$n$ stacks**

$$\text{pop}_k([s_1 \cdots s_m]_k) = [s_2 \cdots s_m]_k$$
Higher-Order Pushdown Automata

Let $\Gamma$ be a stack alphabet. $S_n^\Gamma$ is the set of order-$n$ stacks:

$$S_0^\Gamma = \Gamma \quad \quad S_{k+1}^\Gamma = \{[s_1 \cdots s_m]_{k+1} \mid s_1, \ldots, s_m \in S_k^\Gamma\}.$$ 

Operations on order-$n$ stacks

$$\text{pop}_k([s_1 \cdots s_m]_k) = [s_2 \cdots s_m]_k$$
$$\text{pop}_k([s_1 \cdots s_m]_n) = [\text{pop}_k(s_1)s_2 \cdots s_m]_k \quad n > k$$
Higher-Order Pushdown Automata

Let $\Gamma$ be a stack alphabet. $S_\Gamma^n$ is the set of order-$n$ stacks:

$$S_\Gamma^0 = \Gamma$$
$$S_\Gamma^{k+1} = \{[s_1 \cdots s_m]_{k+1} \mid s_1, \ldots, s_m \in S_\Gamma^k\}.$$

Operations on order-$n$ stacks

$$\text{pop}_k([s_1 \cdots s_m]_k) = [s_2 \cdots s_m]_k$$
$$\text{pop}_k([s_1 \cdots s_m]_n) = [\text{pop}_k(s_1)s_2 \cdots s_m]_k \quad n > k$$
$$\text{push}_k([s_1 \cdots s_m]_k) = [s_1s_1 \cdots s_m]_k$$
Higher-Order Pushdown Automata

Let $\Gamma$ be a stack alphabet. $S_n^\Gamma$ is the set of order-$n$ stacks:

$$S_0^\Gamma = \Gamma \quad \quad S_{k+1}^\Gamma = \{[s_1 \cdots s_m]_{k+1} \mid s_1, \ldots, s_m \in S_k\}.$$ 

Operations on order-$n$ stacks

- $\text{pop}_k([s_1 \cdots s_m]_k) = [s_2 \cdots s_m]_k$
- $\text{pop}_k([s_1 \cdots s_m]_n) = [\text{pop}_k(s_1)s_2 \cdots s_m]_k \quad n > k$
- $\text{push}_k([s_1 \cdots s_m]_k) = [s_1s_1 \cdots s_m]_k$
- $\text{push}_k([s_1 \cdots s_m]_n) = [\text{push}_k(s_1)s_2 \cdots s_m]_k \quad n > k$
Higher-Order Pushdown Automata

Let \( \Gamma \) be a stack alphabet. 
\( S^n_\Gamma \) is the set of order-\( n \) stacks:

\[
S^n_\Gamma = \Gamma \quad \text{for } n \leq 1 \quad \text{and} \quad S^n_{k+1} = \{ [s_1 \cdots s_m]_{k+1} \mid s_1, \ldots, s_m \in S^n_k \}.
\]

### Operations on order-\( n \) stacks

- \( \text{pop}_k([s_1 \cdots s_m]_k) = [s_2 \cdots s_m]_k \)
- \( \text{pop}_k([s_1 \cdots s_m]_n) = [\text{pop}_k(s_1)s_2 \cdots s_m]_k \quad \text{for } n > k \)
- \( \text{push}_k([s_1 \cdots s_m]_k) = [s_1 s_1 \cdots s_m]_k \)
- \( \text{push}_k([s_1 \cdots s_m]_n) = [\text{push}_k(s_1)s_2 \cdots s_m]_k \quad \text{for } n > k \)
- \( \text{rew}_\gamma([\gamma_1 \cdots \gamma_m]_1) = [\gamma_2 \cdots \gamma_m]_1 \)
Higher-Order Pushdown Automata

Let $\Gamma$ be a stack alphabet.

$S_n^\Gamma$ is the set of order-$n$ stacks:

$$S_0^\Gamma = \Gamma, \quad S_{k+1}^\Gamma = \{[s_1 \cdots s_m]_{k+1} \mid s_1, \ldots, s_m \in S_k^\Gamma\}.$$ 

### Operations on order-$n$ stacks

- $\text{pop}_k([s_1 \cdots s_m]_k) = [s_2 \cdots s_m]_k$
- $\text{pop}_k([s_1 \cdots s_m]_n) = [\text{pop}_k(s_1)s_2 \cdots s_m]_k \quad n > k$
- $\text{push}_k([s_1 \cdots s_m]_k) = [s_1s_1 \cdots s_m]_k$
- $\text{push}_k([s_1 \cdots s_m]_n) = [\text{push}_k(s_1)s_2 \cdots s_m]_k \quad n > k$
- $\text{rew}_\gamma([\gamma_1 \cdots \gamma_m]_1) = [\gamma_\gamma2 \cdots \gamma_m]_1$
- $\text{rew}_\gamma([s_1 \cdots s_m]_n) = [\text{rew}_\gamma(s_1)s_2 \cdots s_m]_n \quad n > 1$
Higher-Order Pushdown Automata

Let $\Gamma = \{ \bot, A_0, \ldots, A_k \}$. $\bot$: Initial stack symbol.
Higher-Order Pushdown Automata

Let $\Gamma = \{\bot, A_0, \ldots, A_k\}$. $\bot$: Initial stack symbol.

Order-1 Pushdown Automaton
Higher-Order Pushdown Automata

Let $\Gamma = \{ \perp, A_0, \ldots, A_k \}$. $\perp$: Initial stack symbol.

Order-1 Pushdown Automaton

$A_0, a, \text{pop}_1$

$\perp, \varepsilon, \text{push}_1 \text{rew}_{A_k}$

$\perp, \varepsilon, \text{pop}_1$

$A_i, \varepsilon, \text{rew}_{A_{i-1}} \text{push}_1$

Accepts $t a_2 k u$.  

Order-2 Pushdown Automaton

$A_0, a, \text{pop}_1$

$\perp, \varepsilon, \text{push}_1 \text{rew}_{A_k}$

$\perp, \varepsilon, \text{pop}_1$

$A_i, \varepsilon, \text{rew}_{A_{i-1}} \text{push}_1$

Accepts $t a_2 k u$.  

Georg Zetzsche (LSV Cachan)
Higher-Order Pushdown Automata

Let $\Gamma = \{\bot, A_0, \ldots, A_k\}$. $\bot$: Initial stack symbol.

Order-1 Pushdown Automaton

Order-2 Pushdown Automaton

Accepts $\{a^{2^k}\}$. 
Higher-Order Pushdown Automata

Let $\Gamma = \{\bot, A_0, \ldots, A_k\}$. $\bot$: Initial stack symbol.

**Order-1 Pushdown Automaton**

\[
A_0, a, \text{pop}_1 \\
\bot, \varepsilon, \text{push}_1 \text{ rew}_{A_k} \\
\bot, \varepsilon, \text{pop}_1 \\
A_i, \varepsilon, \text{rew}_{A_{i-1}} \text{ push}_1 \\
\]

Accepts $\{a^{2^k}\}$.

**Order-2 Pushdown Automaton**

\[
A_0, \varepsilon, \text{pop}_1 \text{ push}_2 \\
\bot, \varepsilon, \text{push}_1 \text{ rew}_{A_k} \\
\bot, \varepsilon, \text{pop}_2 \\
A_0, \varepsilon, \text{pop}_1 \text{ push}_2 \\
A_i, \varepsilon, \text{rew}_{A_{i-1}} \text{ push}_1 \\
\]

Georg Zetzsche (LSV Cachan)
Higher-Order Pushdown Automata

Let $\Gamma = \{\bot, A_0, \ldots, A_k\}$. $\bot$: Initial stack symbol.

**Order-1 Pushdown Automaton**

$A_0, a, \text{pop}_1$

$\bot, \varepsilon, \text{push}_1 \text{ rew}_{A_k}$

$\bot, \varepsilon, \text{pop}_1$

Accepts $\{a^{2^k}\}$.

$A_i, \varepsilon, \text{rew}_{A_{i-1}} \text{ push}_1$

**Order-2 Pushdown Automaton**

$\bot, a, \text{pop}_2$

$\bot, \varepsilon, \text{push}_1 \text{ rew}_{A_k}$

$\bot, \varepsilon, \text{pop}_2$

Accepts $\{a^{2^{2^k}}\}$.

$A_0, \varepsilon, \text{pop}_1 \text{ push}_2$

$A_i, \varepsilon, \text{rew}_{A_{i-1}} \text{ push}_1$
Complexity

Descriptional complexity

- What is the size of an automaton for $L\downarrow$?
- How expensive the construction?

Inclusion problem

For models $M$ and $N$ the downward closure inclusion problem $M \subseteq^* L \subseteq^* N$ is the following:

Given: Language $K$, $L$ described in $M$ and $N$, respectively.

Question: Does $K \subseteq^* L$?
Complexity

Descriptive complexity
- What is the size of an automaton for $L\downarrow$?
- How expensive the construction?

Inclusion problem
For models $\mathcal{M}$ and $\mathcal{N}$ the downward closure inclusion problem $\mathcal{M} \subseteq \downarrow \mathcal{N}$ is the following:

Given: Language $K$, $L$ described in $\mathcal{M}$ and $\mathcal{N}$, respectively.
Question: Does $K\downarrow \subseteq L\downarrow$?
Complexity

Descriptive complexity
- What is the size of an automaton for $L\downarrow$?
- How expensive the construction?

Inclusion problem
For models $\mathcal{M}$ and $\mathcal{N}$ the downward closure inclusion problem $\mathcal{M} \subseteq \downarrow \mathcal{N}$ is the following:

Given: Language $K$, $L$ described in $\mathcal{M}$ and $\mathcal{N}$, respectively.
Question: Does $K\downarrow \subseteq L\downarrow$?

Equality problem
For models $\mathcal{M}$ and $\mathcal{N}$ the downward closure inclusion problem $\mathcal{M} = \downarrow \mathcal{N}$ is the following:

Given: Language $K$, $L$ described in $\mathcal{M}$ and $\mathcal{N}$, respectively.
Question: Does $K\downarrow = L\downarrow$?
Witnesses

Suppose we have an NFA for the downward closure.
Witnesses

Suppose we have an NFA for the downward closure. “Short witness”:

**Proposition (Bachmeier, Luttenberger, Schlund 2015)**

*If \( A \) is an NFA and \( K \downarrow \subseteq L(A) \downarrow \), then there exists a \( w \in K \downarrow \setminus L(A) \downarrow \) with \(|w| \leq |A| + 1\).*

Georg Zetzsche (LSV Cachan)
Witnesses

Suppose we have an NFA for the downward closure. “Short witness”:

Proposition (Bachmeier, Luttenberger, Schlund 2015)

*If* \( A \) *is an NFA and* \( K \downarrow \subseteq L(A) \downarrow \), *then there exists a* \( w \in K \downarrow \setminus L(A) \downarrow \) *with* \( |w| \leq |A| + 1 \).

- Suppose \( A \) has parallel \( \varepsilon \)-edges
Witnesses

Suppose we have an NFA for the downward closure. “Short witness”:

Proposition (Bachmeier, Luttenberger, Schlund 2015)

If \( \mathcal{A} \) is an NFA and \( K \downarrow \supseteq L(\mathcal{A}) \downarrow \), then there exists a \( w \in K \downarrow \setminus L(\mathcal{A}) \downarrow \) with \( |w| \leq |\mathcal{A}| + 1 \).

- Suppose \( \mathcal{A} \) has parallel \( \varepsilon \)-edges
- For an input word \( w = x_1 \cdots x_n \), consider the sets \( Q_i \) of words reachable by \( x_1 \cdots x_i \). Then \( Q_0 \supseteq Q_1 \supseteq \cdots \)
Witnesses

Suppose we have an NFA for the downward closure. “Short witness”:

Proposition (Bachmeier, Luttenberger, Schlund 2015)

If $A$ is an NFA and $K \downarrow \subseteq L(A) \downarrow$, then there exists a $w \in K \downarrow \setminus L(A) \downarrow$ with $|w| \leq |A| + 1$.

- Suppose $A$ has parallel $\varepsilon$-edges
- For an input word $w = x_1 \cdots x_n$, consider the sets $Q_i$ of words reachable by $x_1 \cdots x_i$. Then $Q_0 \supseteq Q_1 \supseteq \cdots$
- Hence, if $n > |A| + 1$, then we can remove some symbol from $w$
Witnesses

Suppose we have an NFA for the downward closure. “Short witness”:

**Proposition (Bachmeier, Luttenberger, Schlund 2015)**

If $\mathcal{A}$ is an NFA and $K \downarrow \subseteq L(\mathcal{A}) \downarrow$, then there exists a $w \in K \downarrow \backslash L(\mathcal{A}) \downarrow$ with $|w| \leq |\mathcal{A}| + 1$.

- Suppose $\mathcal{A}$ has parallel $\varepsilon$-edges
- For an input word $w = x_1 \cdots x_n$, consider the sets $Q_i$ of words reachable by $x_1 \cdots x_i$. Then $Q_0 \supseteq Q_1 \supseteq \cdots$
- Hence, if $n > |\mathcal{A}| + 1$, then we can remove some symbol from $w$

Suppose we have no good upper bound on an NFA.

**Ideal length**

For $I = \varepsilon \cdot 0 \cdot x_1, \varepsilon \cdot u \cdot \varepsilon \cdot 0 \cdot \varepsilon \cdot x_2, \cdots, \varepsilon \cdot u \cdot \varepsilon \cdot 0 \cdot x_n, \varepsilon \cdot u$, the length $|I|$ of $I$ is the smallest $n$ such that it can be written in this form.

**Measure for languages:**

$|L| = \min \max_{i=1}^{k} |I_i|$, where $L = I_1 \cdot \varepsilon \cdots \cdot I_k$ for ideals $I_1, \ldots, I_k$. 

Georg Zetzsche (LSV Cachan)
Witnesses

Suppose we have an NFA for the downward closure. “Short witness”:

**Proposition (Bachmeier, Luttenberger, Schlund 2015)**

*If \( \mathcal{A} \) is an NFA and \( K \downarrow \subseteq L(\mathcal{A}) \downarrow \), then there exists a \( w \in K \downarrow \setminus L(\mathcal{A}) \downarrow \) with \( |w| \leq |\mathcal{A}| + 1 \).*

- Suppose \( \mathcal{A} \) has parallel \( \varepsilon \)-edges
- For an input word \( w = x_1 \cdots x_n \), consider the sets \( Q_i \) of words reachable by \( x_1 \cdots x_i \). Then \( Q_0 \supseteq Q_1 \supseteq \cdots \)
- Hence, if \( n > |\mathcal{A}| + 1 \), then we can remove some symbol from \( w \)

Suppose we have no good upper bound on an NFA.

**Ideal length**

For \( I = Y_0^* \{ x_1, \varepsilon \} Y_1^* \cdots \{ x_n, \varepsilon \} Y_n^* \), the *length* \( |I| \) of \( I \) is the smallest \( n \) such that it can be written in this form.
Witnesses

Suppose we have an NFA for the downward closure. “Short witness”:

Proposition (Bachmeier, Luttenberger, Schlund 2015)

If \( \mathcal{A} \) is an NFA and \( K \downarrow \subseteq L(\mathcal{A}) \downarrow \), then there exists a \( w \in K \downarrow \setminus L(\mathcal{A}) \downarrow \) with \( |w| \leq |A| + 1 \).

- Suppose \( \mathcal{A} \) has parallel \( \varepsilon \)-edges
- For an input word \( w = x_1 \cdots x_n \), consider the sets \( Q_i \) of words reachable by \( x_1 \cdots x_i \). Then \( Q_0 \supseteq Q_1 \supseteq \cdots \)
- Hence, if \( n > |A| + 1 \), then we can remove some symbol from \( w \)

Suppose we have no good upper bound on an NFA.

Ideal length

For \( I = Y_0^* \{ x_1, \varepsilon \} Y_1^* \cdots \{ x_n, \varepsilon \} Y_n^* \), the length \( |I| \) of \( I \) is the smallest \( n \) such that it can be written in this form. Measure for languages:

\[
|L| = \min \left\{ \max_j |I_j| : \ L \downarrow = I_1 \cup \cdots \cup I_k \text{ for ideals } I_1, \ldots, I_k \right\}
\]
Pumping

Putting a bound on $|L|$ amounts to proving a pumping lemma: $|L| \leq m$ if and only if for every $w \in L$, there is an ideal $I$ such that $|I| \leq m$ and $w \in I \subseteq L\downarrow$. 
Pumping

Putting a bound on $|L|$ amounts to proving a pumping lemma:
$|L| \leq m$ if and only if for every $w \in L$, there is an ideal $I$ such that $|I| \leq m$ and $w \in I \subseteq L\downarrow$.

Proposition (Ideal witness)

Let $I = Y_0^*\{x_1, \varepsilon\}Y_1^*\cdots\{x_n, \varepsilon\}Y_n^*$. Then the following are equivalent:

1. $I \subseteq L\downarrow$.
2. $w_{Y_0}^m x_1 w_{Y_1}^m \cdots x_n w_{Y_n}^m \in L\downarrow$ for every $m \geq |L| + 1$.
3. $w_{Y_0}^m x_1 w_{Y_1}^m \cdots x_n w_{Y_n}^m \in L\downarrow$ for some $m \geq |L| + 1$. 
Pumping

Putting a bound on \(|L|\) amounts to proving a pumping lemma: \(|L| \leq m\) if and only if for every \(w \in L\), there is an ideal \(I\) such that \(|I| \leq m\) and \(w \in I \subseteq L^\downarrow\).

Proposition (Ideal witness)

Let \(I = Y_0^* \{x_1, \varepsilon\} Y_1^* \cdots \{x_n, \varepsilon\} Y_n^*\). Then the following are equivalent:

1. \(I \subseteq L^\downarrow\).
2. \(w_{Y_0}^m x_1 w_{Y_1}^m \cdots x_n w_{Y_n}^m \in L^\downarrow\) for every \(m \geq |L| + 1\).
3. \(w_{Y_0}^m x_1 w_{Y_1}^m \cdots x_n w_{Y_n}^m \in L^\downarrow\) for some \(m \geq |L| + 1\).

Strategy for \(K^\downarrow \subseteq L^\downarrow\)

- Suppose \(|K|\) is polynomial and \(|L|\) exponential.
- Guess an ideal \(I\) of length \(\leq |K|\) and verify \(I \subseteq K, I \not\subseteq L\).
- Represent witness above succinctly.
Sometimes, we have a small bound on $|L|$ but only a large bound on $|K|$. 

Proposition (Small alphabet witness)

Let $K, L \subseteq X$. If $K \subseteq L$, then there exists a $w \in K$ with $|w| \leq |L|$. This yields existence of small witnesses for fixed alphabets.

Turn every ideal into an ordered DFA: no cycles except self-loops

Then prove the following:

Lemma

If $w \in X^*$ with $|w| \geq |X|$, then $w$ has a position at which every ordered $n$-state DFA cycles.

Induction yields length bound for subwords with less than $|X|$ symbols

Decompose $w$ into at least $n$ factors each of which sees all of $X$.

Each DFA has to repeat a state, hence cycle on the last letter of $w$.
Sometimes, we have a small bound on $|L|$ but only a large bound on $|K|$.

**Proposition (Small alphabet witness)**

Let $K, L \subseteq X^*$. If $K \Downarrow \subseteq L\Downarrow$, then there exists a $w \in K\Downarrow \setminus L\Downarrow$ with $|w| \leq |X| \cdot (|L| + 1)^{|X|}$.

- This yields existence of small witnesses for fixed alphabets.
Sometimes, we have a small bound on $|L|$ but only a large bound on $|K|$.

**Proposition (Small alphabet witness)**

Let $K, L \subseteq X^*$. If $K \downarrow \subseteq L \downarrow$, then there exists a $w \in K \downarrow \setminus L \downarrow$ with $|w| \leq |X| \cdot (|L| + 1)^{|X|}$.

- This yields existence of small witnesses for fixed alphabets.
- Turn every ideal into an ordered DFA: no cycles except self-loops.
Sometimes, we have a small bound on $|L|$ but only a large bound on $|K|$.

**Proposition (Small alphabet witness)**

Let $K, L \subseteq X^*$. If $K \downarrow \subseteq L \downarrow$, then there exists a $w \in K \downarrow \backslash L \downarrow$ with $|w| \leq |X| \cdot (|L| + 1)^{|X|}$.

- This yields existence of small witnesses for fixed alphabets.
- Turn every ideal into an *ordered DFA*: no cycles except self-loops.
- Then prove the following:

**Lemma**

*If* $w \in X^*$ *with* $|w| > |X| \cdot (n - 1)^{|X|}$, *then* $w$ *has a position at which every* ordered $n$-*state DFA cycles.*
Sometimes, we have a small bound on $|L|$ but only a large bound on $|K|$.

**Proposition (Small alphabet witness)**

Let $K, L \subseteq X^*$. If $K \downarrow \subseteq L \downarrow$, then there exists a $w \in K \downarrow \setminus L \downarrow$ with $|w| \leq |X| \cdot (|L| + 1)^{|X|}$.

- This yields existence of small witnesses for fixed alphabets.
- Turn every ideal into an **ordered DFA**: no cycles except self-loops
- Then prove the following:

**Lemma**

If $w \in X^*$ with $|w| > |X| \cdot (n - 1)^{|X|}$, then $w$ has a position at which every ordered $n$-state DFA cycles.

- Induction yields length bound for subwords with less than $|X|$ symbols
Sometimes, we have a small bound on $|L|$ but only a large bound on $|K|$.

**Proposition (Small alphabet witness)**

Let $K, L \subseteq X^*$. If $K \downarrow \not\subseteq L \downarrow$, then there exists a $w \in K \downarrow \setminus L \downarrow$ with $|w| \leq |X| \cdot (|L| + 1)^{|X|}$.

- This yields existence of small witnesses for fixed alphabets.
- Turn every ideal into an ordered DFA: no cycles except self-loops.
- Then prove the following:

**Lemma**

If $w \in X^*$ with $|w| > |X| \cdot (n - 1)^{|X|}$, then $w$ has a position at which every ordered $n$-state DFA cycles.

- Induction yields length bound for subwords with less than $|X|$ symbols.
- Decompose $w$ into at least $n$ factors each of which sees all of $X$. 
Sometimes, we have a small bound on $|L|$ but only a large bound on $|K|$.

**Proposition (Small alphabet witness)**

Let $K, L \subseteq X^*$. If $K \downarrow \subseteq L\downarrow$, then there exists a $w \in K\downarrow \setminus L\downarrow$ with

$$|w| \leq |X| \cdot (|L| + 1)^{|X|}.$$ 

- This yields existence of small witnesses for fixed alphabets.
- Turn every ideal into an ordered DFA: no cycles except self-loops.
- Then prove the following:

**Lemma**

If $w \in X^*$ with $|w| > |X| \cdot (n - 1)^{|X|}$, then $w$ has a position at which every ordered $n$-state DFA cycles.

- Induction yields length bound for subwords with less than $|X|$ symbols.
- Decompose $w$ into at least $n$ factors each of which sees all of $X$.
- Each DFA has to repeat a state, hence cycle on the last letter of $w$.
Lower bounds

Theorem

There are order-$n$ pushdown automata whose downward closure NFAs are at least $n$-fold exponential.
Lower bounds

**Theorem**

*There are order-\(n\) pushdown automata whose downward closure NFAs are at least \(n\)-fold exponential.*

- An NFA for \(\{w\} \downarrow\) requires \(|w| + 1\) states.
Lower bounds

**Theorem**

There are order-\(n\) pushdown automata whose downward closure NFAs are at least \(n\)-fold exponential.

- An NFA for \(\{w\}\downarrow\) requires \(|w| + 1\) states.
- Examples for \(\{a^{2^k}\}\) and \(\{a^{2^{2^k}}\}\) extend easily to order \(n\).
Lower bounds

**Theorem**

There are order-$n$ pushdown automata whose downward closure NFAs are at least $n$-fold exponential.

- An NFA for $\{w\} \downarrow$ requires $|w| + 1$ states.
- Examples for $\{a^{2^k}\}$ and $\{a^{2^{2^k}}\}$ extend easily to order $n$.

**Theorem**

Under mild conditions on the models $M$ and $N$: Suppose for each $n$ we have a description of $\{a^{t(n)}\}$ in $M$ and $N$ of polynomial size. Then $M \subseteq \downarrow N$ is hard for coNTIME$(t)$.
Lower bounds

Theorem

There are order-\(n\) pushdown automata whose downward closure NFAs are at least \(n\)-fold exponential.

- An NFA for \(\{w\}\downarrow\) requires \(|w| + 1\) states.
- Examples for \(\{a^{2^k}\}\) and \(\{a^{2^2^k}\}\) extend easily to order \(n\).

Theorem

Under mild conditions on the models \(\mathcal{M}\) and \(\mathcal{N}\): Suppose for each \(n\) we have a description of \(\{a^{t(n)}\}\) in \(\mathcal{M}\) and \(\mathcal{N}\) of polynomial size. Then \(\mathcal{M} \subseteq \downarrow \mathcal{N}\) is hard for \(\text{coNTIME}(t)\).

Corollary

The problem \(\text{HOPA}_n \subseteq \downarrow \text{HOPA}_n\) is co-\(n\)-\(\text{NEXP}\)-hard.
<table>
<thead>
<tr>
<th></th>
<th>Ideal</th>
<th>NFA</th>
<th>OCA</th>
<th>RBC&lt;sub&gt;k,r&lt;/sub&gt;</th>
<th>CFG</th>
<th>RBC</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Ideal</strong></td>
<td>∈ L</td>
<td>NL</td>
<td>NL</td>
<td>NL</td>
<td>P</td>
<td>NP</td>
</tr>
</tbody>
</table>
| **NFA**  | NL    | coNP| coNP| coNP             | coNP   | Π<sub>2</sub>
| **OCA**  | NL    | coNP| coNP| coNP             | coNP   | Π<sub>2</sub>
| **RBC<sub>k,r</sub>** | NL | coNP| coNP| coNP             | coNP   | Π<sub>2</sub>
| **CFG**  | P     | coNP| coNP| coNP<sup>†</sup> | coNEXP | coNEXP |
| **RBC**  | coNP  | coNP| coNP| coNP<sup>†</sup> | coNEXP | coNEXP |
Often better complexity than constructing an NFA:

- NFA $\subseteq$ CFG: NFA for the CFG would be exponential, but problem is coNP-complete.

<table>
<thead>
<tr>
<th></th>
<th>Ideal</th>
<th>NFA</th>
<th>OCA</th>
<th>RBC$_{k,r}$</th>
<th>CFG</th>
<th>RBC</th>
</tr>
</thead>
<tbody>
<tr>
<td>Ideal</td>
<td>∈ L</td>
<td>NL</td>
<td>NL</td>
<td>NL</td>
<td>P</td>
<td>NP</td>
</tr>
<tr>
<td>NFA</td>
<td>NL</td>
<td>coNP</td>
<td>coNP</td>
<td>coNP</td>
<td>coNP</td>
<td>$\Pi_2^P$</td>
</tr>
<tr>
<td>OCA</td>
<td>NL</td>
<td>coNP</td>
<td>coNP</td>
<td>coNP</td>
<td>coNP</td>
<td>$\Pi_2^P$</td>
</tr>
<tr>
<td>RBC$_{k,r}$</td>
<td>NL</td>
<td>coNP</td>
<td>coNP</td>
<td>coNP</td>
<td>coNP</td>
<td>$\Pi_2^P$</td>
</tr>
<tr>
<td>CFG</td>
<td>P</td>
<td>coNP</td>
<td>coNP</td>
<td>coNP$^\dagger$</td>
<td>coNEXP</td>
<td>coNEXP</td>
</tr>
<tr>
<td>RBC</td>
<td>coNP</td>
<td>coNP</td>
<td>coNP</td>
<td>coNP$^\dagger$</td>
<td>coNEXP</td>
<td>coNEXP</td>
</tr>
<tr>
<td></td>
<td>Ideal</td>
<td>NFA</td>
<td>OCA</td>
<td>RBC$_{k,r}$</td>
<td>CFG</td>
<td>RBC</td>
</tr>
<tr>
<td>-------</td>
<td>-------</td>
<td>----------</td>
<td>---------</td>
<td>-------------</td>
<td>-----</td>
<td>-----</td>
</tr>
<tr>
<td>Ideal</td>
<td>∈ L</td>
<td>NL</td>
<td>NL</td>
<td>NL</td>
<td>P</td>
<td>NP</td>
</tr>
<tr>
<td>NFA</td>
<td>NL</td>
<td>coNP</td>
<td>coNP</td>
<td>coNP</td>
<td>coNP</td>
<td>Π$_2^P$</td>
</tr>
<tr>
<td>OCA</td>
<td>NL</td>
<td>coNP</td>
<td>coNP</td>
<td>coNP</td>
<td>coNP</td>
<td>Π$_2^P$</td>
</tr>
<tr>
<td>RBC$_{k,r}$</td>
<td>NL</td>
<td>coNP</td>
<td>coNP</td>
<td>coNP</td>
<td>coNP</td>
<td>Π$_2^P$</td>
</tr>
<tr>
<td>CFG</td>
<td>P</td>
<td>coNP</td>
<td>coNP</td>
<td>coNP$^*$</td>
<td>coNEXP</td>
<td>coNEXP</td>
</tr>
<tr>
<td>RBC</td>
<td>coNP</td>
<td>coNP</td>
<td>coNP</td>
<td>coNP$^*$</td>
<td>coNEXP</td>
<td>coNEXP</td>
</tr>
</tbody>
</table>

Often better complexity than constructing an NFA:

- NFA $\subseteq$ CFG: NFA for the CFG would be exponential, but problem is coNP-complete.
- HOPA$_2$ $\subseteq$ NFA is EXPTIME-complete: Using “short witness”. Hardness inherited from emptiness.
<table>
<thead>
<tr>
<th></th>
<th>Ideal</th>
<th>NFA</th>
<th>OCA</th>
<th>RBC_{k,r}</th>
<th>CFG</th>
<th>RBC</th>
</tr>
</thead>
<tbody>
<tr>
<td>Ideal</td>
<td>∈ L</td>
<td>NL</td>
<td>NL</td>
<td>NL</td>
<td>P</td>
<td>NP</td>
</tr>
<tr>
<td>NFA</td>
<td>NL</td>
<td>coNP</td>
<td>coNP</td>
<td>coNP</td>
<td>coNP</td>
<td>Π_{2}^{P}</td>
</tr>
<tr>
<td>OCA</td>
<td>NL</td>
<td>coNP</td>
<td>coNP</td>
<td>coNP</td>
<td>coNP</td>
<td>Π_{2}^{P}</td>
</tr>
<tr>
<td>RBC_{k,r}</td>
<td>NL</td>
<td>coNP</td>
<td>coNP</td>
<td>coNP</td>
<td>coNP</td>
<td>Π_{2}^{P}</td>
</tr>
<tr>
<td>CFG</td>
<td>P</td>
<td>coNP</td>
<td>coNP</td>
<td>coNP\uparrow</td>
<td>coNEXP</td>
<td>coNEXP</td>
</tr>
<tr>
<td>RBC</td>
<td>coNP</td>
<td>coNP</td>
<td>coNP</td>
<td>coNP\uparrow</td>
<td>coNEXP</td>
<td>coNEXP</td>
</tr>
</tbody>
</table>

- Often better complexity than constructing an NFA:
  - NFA $\subseteq$ CFG: NFA for the CFG would be exponential, but problem is coNP-complete.
  - HOPA_{2} $\subseteq$ NFA is EXPTIME-complete: Using “short witness”. Hardness inherited from emptiness.
- Pumping lemmas may give us bounds on ideal lengths
<table>
<thead>
<tr>
<th></th>
<th>Ideal</th>
<th>NFA</th>
<th>OCA</th>
<th>RBC_{k,r}</th>
<th>CFG</th>
<th>RBC</th>
</tr>
</thead>
<tbody>
<tr>
<td>Ideal</td>
<td>∈ L</td>
<td>NL</td>
<td>NL</td>
<td>NL</td>
<td>P</td>
<td>NP</td>
</tr>
<tr>
<td>NFA</td>
<td>NL</td>
<td>coNP</td>
<td>coNP</td>
<td>coNP</td>
<td>coNP</td>
<td>Π^P_2</td>
</tr>
<tr>
<td>OCA</td>
<td>NL</td>
<td>coNP</td>
<td>coNP</td>
<td>coNP</td>
<td>coNP</td>
<td>Π^P_2</td>
</tr>
<tr>
<td>RBC_{k,r}</td>
<td>NL</td>
<td>coNP</td>
<td>coNP</td>
<td>coNP</td>
<td>coNP</td>
<td>Π^P_2</td>
</tr>
<tr>
<td>CFG</td>
<td>P</td>
<td>coNP</td>
<td>coNP</td>
<td>coNP \dagger</td>
<td>coNEXP</td>
<td>coNEXP</td>
</tr>
<tr>
<td>RBC</td>
<td>coNP</td>
<td>coNP</td>
<td>coNP</td>
<td>coNP \dagger</td>
<td>coNEXP</td>
<td>coNEXP</td>
</tr>
</tbody>
</table>

- Often better complexity than constructing an NFA:
  - NFA ⊆↓ CFG: NFA for the CFG would be exponential, but problem is coNP-complete.
  - HOPA_2 ⊆↓ NFA is EXPTIME-complete: Using “short witness”. Hardness inherited from emptiness.
- Pumping lemmas may give us bounds on ideal lengths
- The problem HOPA_n ⊆↓ HOPA_n is co-n-NEXP-hard.