An approach to computing downward closures

Georg Zetzsche

Technische Universität Kaiserslautern

ICALP 2015
System Observer

Downward Closures $u \leq v$: $u$ is a subsequence of $v$

Observer sees precisely $u \leq v$.
System

Observer

LOSSY CHANNEL

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

Observer sees precisely

Georg Zetzsche (TU KL)
Computing Downward Closures
ICALP 2015 2 / 16
Downward Closures

System Observer

LOSSY CHANNEL

observer sees precisely $L^\delta_t u = P_X \delta v$

$u$ is a subsequence of $v$

aabcbbacbbaaab

abbbcbca

aabcbbacbbaaab
System Observer
LOSSY
CHANNEL

Downward Closures

- $u \trianglelefteq v$: $u$ is a subsequence of $v$
- $L \downarrow = \{u \in X^* \mid \exists v \in L: u \trianglelefteq v\}$
- 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 ($K\subseteq L\downarrow$)
- Ordinary inclusion almost always undecidable!
- Which actions occur arbitrarily often? ($a \in L\downarrow$)
- Is $a$ ever executed after $b$? ($ab \in L\downarrow$)
- Can the system run arbitrarily long? ($L\downarrow$ infinite)

Problem

Finite automaton for $L\downarrow$ exists for every $L$.

How can we compute it?
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 ($K \subseteq L\downarrow$)
- Ordinary inclusion almost always undecidable!
- Which actions occur arbitrarily often? ($a \subseteq L\downarrow$)
- Is $a$ ever executed after $b$? ($ab \not \in L\downarrow$)
- Can the system run arbitrarily long? ($L\downarrow$ infinite)

Problem

Finite automaton for $L\downarrow$ exists for every $L$.
How can we compute it?
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? (\( a^* \subseteq L \downarrow \))
- Is \( a \) ever executed after \( b \)? (\( ab \in L \downarrow \))

Georg Zetzsche (TU KL)
Computing Downward Closures
ICALP 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? ($a^* \subseteq L\downarrow$)
- Is $a$ ever executed after $b$? ($ab \in L\downarrow$)
- 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? ($a^* \subseteq L \downarrow$)
- Is $a$ ever executed after $b$? ($ab \in L \downarrow$)
- Can the system run arbitrarily long? ($L \downarrow$ infinite)

## Problem

- Finite automaton for $L \downarrow$ exists for every $L$.
- How can we compute it?
Negative results

Theorem (Gruber, Holzer, Kutrib 2009)

*Downward closures are not computable when infinity or emptiness are undecidable.*

Theorem (Mayr 2003)

*The reachability set of lossy channel systems is not computable.*
Positive results

Theorem (van Leeuwen 1978/Courcelle 1991)
Downward closures are computable for context-free languages.

Theorem (Abdulla, Boasson, Bouajjani, ICALP 2001)
Downward closures are computable for context-free FIFO rewriting systems/0L-systems.

Context-free rules
\[ A \rightarrow w, \text{ applied as: } Au \rightarrow uw \]

Theorem (Habermehl, Meyer, Wimmel, ICALP 2010)
Downward closures are computable for Petri net languages.

Theorem (Z., STACS 2015)
Downward closures are computable for stacked counter automata.

Weak form of stack nesting
Adding Counters
Positive results

Theorem (van Leeuwen 1978/Courcelle 1991)

*Downward closures are computable for context-free languages.*
Positive results

Theorem (van Leeuwen 1978/Courcelle 1991)

*Downward closures are computable for context-free languages.*

Theorem (Abdulla, Boasson, Bouajjani, ICALP 2001)

*Downward closures are computable for context-free FIFO rewriting systems/0L-systems.*
Positive results

Theorem (van Leeuwen 1978/Courcelle 1991)

*Downward closures are computable for context-free languages.*

Theorem (Abdulla, Boasson, Bouajjani, ICALP 2001)

*Downward closures are computable for context-free FIFO rewriting systems/0L-systems.*

Context-free rules $A \rightarrow w$, applied as: $Au \Rightarrow uw$
## Positive results

### Theorem (van Leeuwen 1978/Courcelle 1991)

*Downward closures are computable for context-free languages.*

### Theorem (Abdulla, Boasson, Bouajjani, ICALP 2001)

*Downward closures are computable for context-free FIFO rewriting systems/0L-systems.*

Context-free rules $A \rightarrow w$, applied as: $Au \Rightarrow uw$

### Theorem (Habermehl, Meyer, Wimmel, ICALP 2010)

*Downward closures are computable for Petri net languages.*
Positive results

Theorem (van Leeuwen 1978/Courcelle 1991)

*Downward closures are computable for context-free languages.*

Theorem (Abdulla, Boasson, Bouajjani, ICALP 2001)

*Downward closures are computable for context-free FIFO rewriting systems/0L-systems.*

Context-free rules $A \rightarrow w$, applied as: $Au \Rightarrow uw$

Theorem (Habermehl, Meyer, Wimmel, ICALP 2010)

*Downward closures are computable for Petri net languages.*

Theorem (Z., STACS 2015)

*Downward closures are computable for stacked counter automata.*
## Positive results

| Theorem (van Leeuwen 1978/Courcelle 1991) | Downward closures are computable for context-free languages. |
| Theorem (Abdulla, Boasson, Bouajjani, ICALP 2001) | Downward closures are computable for context-free FIFO rewriting systems/0L-systems. |
| Context-free rules $A \rightarrow w$, applied as: $Au \Rightarrow uw$ |
| Theorem (Habermehl, Meyer, Wimmel, ICALP 2010) | Downward closures are computable for Petri net languages. |
| Theorem (Z., STACS 2015) | Downward closures are computable for stacked counter automata. |

- Weak form of stack nesting
- Adding Counters
A general approach

Example (Transducer)

\[ q_0 \xrightarrow{\varepsilon} q_1 \xrightarrow{a|a, b|b} q_2 \]

\[ q_1 \xrightarrow{\varepsilon|\#} q_2 \]

\[ q_0 \xrightarrow{\varepsilon|\#} q_1 \]

Definition

Rational transduction: set of pairs given by a finite state transducer. For rational transduction \( T \), let \( T(L) \) for language \( L \):

\[ T(L) = \{ x, y | (p, x, y, q) \in T \} \]
A general approach

Example (Transducer)

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

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

Example (Transducer)

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

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

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

Theorem

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 $L\downarrow$?
Theorem (Jullien 1969, Abdulla et. al. 2004)

Every language $L \downarrow$ can be written as a finite union of sets of the form

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

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

“Simple Regular Languages”
Theorem (Jullien 1969, Abdulla et. al. 2004)

Every language \( L \subseteq \Sigma \) can be written as a finite union of sets of the form

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

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

“Simple Regular Languages” ← Ideal decomposition!
Theorem (Jullien 1969, Abdulla et. al. 2004)

Every language $L$ can be written as a finite union of sets of the form

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

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

“Simple Regular Languages” ← Ideal decomposition!

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 sets of the form

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

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

“Simple Regular Languages” ← Ideal decomposition!

Algorithm

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

Every language $L$ can be written as a finite union of sets of the form

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

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

“Simple Regular Languages” ← Ideal decomposition!

Algorithm

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

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

Observation

$L \downarrow$ is in $C$:

- $(x, \varepsilon)$
- $(x, x)$
Every language $L$ can be written as a finite union of sets of the form

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

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

“Simple Regular Languages” ← Ideal decomposition!

Algorithm

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

Enumerate simple regular languages $R$.

Decide whether $L \downarrow = R$:

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

Observation

$L \downarrow$ is in $\mathcal{C}$:

$$(x, \varepsilon)$$

$$(x, x)$$
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 $(abc)^*$.

\[\text{abc abc abc abc abc abc}\]
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 $(abc)^*$.

\[
\begin{align*}
abc & \quad abc & \quad abc & \quad abc & \quad abc & \quad abc \\
& \quad bacca
\end{align*}
\]
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 $(abc)^*$.

\[ abc \ abc \ ab\ abc \ abc \]
\[ bacca \]
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 $(abc)^*$.

\[ abc \quad abc \quad abc \quad abc \quad abc \]
\[ bacca \]

Transduction $T$

\[ y_0 | a_0 \]
\[ q_0 \]
\[ \longrightarrow \]
\[ x_1 | \varepsilon \]
\[ q_1 \]
\[ \longrightarrow \]
\[ y_1 | a_1 \]
\[ y_2 | a_2 \]
\[ \cdots \]
\[ y_n | a_n \]
\[ q_n \]

$y_i$: 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 \( T \)

\[
\begin{aligned}
y_0 | a_0 & \quad y_1 | a_1 & \quad y_n | a_n \\
q_0 & \quad q_1 & \quad q_n \\
x_1 | \varepsilon & \quad x_2 | \varepsilon & \quad x_n | \varepsilon
\end{aligned}
\]

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

\[
T( L\downarrow )\downarrow = a_0^* \cdots a_n^* \quad \text{iff} \quad Y_0^* \{ x_1, \varepsilon \} Y_1^* \cdots \{ x_n, \varepsilon \} Y_n^* \subseteq L\downarrow
\]
New algorithm for each known positive case

Context-free grammars and stacked counter automata:

Corollary

*If C is a full trio and has effectively semilinear Parikh images, then downward closures are computable for C.*
New algorithm for each known positive case

Context-free grammars and stacked counter automata:

**Corollary**

*If $C$ is a full trio and has effectively semilinear Parikh images, then downward closures are computable for $C$.*

Petri net languages ⇝ boundedness with one inhibitor arc (Czerwiński, Martens 2015), decidable by (Bonnet et. al. 2012)
New algorithm for each known positive case

Context-free grammars and stacked counter automata:

**Corollary**

*If \( C \) is a full trio and has effectively semilinear Parikh images, then downward closures are computable for \( C \).*

Petri net languages \( \rightsquigarrow \) boundedness with one inhibitor arc (Czerwiński, Martens 2015), decidable by (Bonnet et. al. 2012)

**Theorem**

*Downward closures are computable for matrix languages.*

Natural generalization of context-free and Petri net languages.
New algorithm for each known positive case

Context-free grammars and stacked counter automata:

**Corollary**

*If C is a full trio and has effectively semilinear Parikh images, then downward closures are computable for C.*

Petri net languages \(\leadsto\) boundedness with one inhibitor arc (Czerwiński, Martens 2015), decidable by (Bonnet et. al. 2012)

**Theorem**

*Downward closures are computable for matrix languages.*

Natural generalization of context-free and Petri net languages.

**Theorem**

*Downward closures are computable for indexed languages.*

(Generalize 0L-systems)
Indexed Grammars

Idea: Each nonterminal carries a stack.
Indexed Grammars

Indexed Grammars

Idea: Each nonterminal carries a stack.
Tuple $G = (N, T, I, P, S)$, where
- $N, T, I$ are nonterminal, terminal, index alphabet,
- $S \in N$ start symbol
Indexed Grammars

Indexed Grammars

Idea: Each nonterminal carries a stack.
Tuple $G = (N, T, I, P, S)$, where

- $N, T, I$ are nonterminal, terminal, index alphabet,
- $S \in N$ start symbol
- Productions $P$ of the form
  - $A \rightarrow Bf$, push index ($f \in I$)
  - $Af \rightarrow B$, pop index ($f \in I$)
  - $A \rightarrow uBv$, generate terminals ($u, v \in T^*$)
  - $A \rightarrow BC$, split and duplicate index word
  - $A \rightarrow w$, generate only terminals ($w \in T^*$)
Indexed Grammars

Indexed Grammars

Idea: Each nonterminal carries a stack.
Tuple $G = (N, T, I, P, S)$, where

- $N, T, I$ are nonterminal, terminal, index alphabet,
- $S \in N$ start symbol
- Productions $P$ of the form
  - $A \rightarrow Bf$, push index ($f \in I$)
  - $Af \rightarrow B$, pop index ($f \in I$)
  - $A \rightarrow uBv$, generate terminals ($u, v \in T^*$)
  - $A \rightarrow BC$, split and duplicate index word
  - $A \rightarrow w$, generate only terminals ($w \in T^*$)

$S \rightarrow Sf, \quad S \rightarrow Sg, \quad S \rightarrow UU, \quad U \rightarrow \varepsilon,$

$Uf \rightarrow A, \quad Ug \rightarrow B, \quad A \rightarrow Ua, \quad B \rightarrow Ub.$

$N = \{S, T, A, B\}, I = \{f, g\}, T = \{a, b\}.$
Indexed Grammars

**Indexed Grammars**

Idea: Each nonterminal carries a stack.

Tuple \( G = (N, T, I, P, S) \), where

- \( N, T, I \) are nonterminal, terminal, index alphabet,
- \( S \in N \) start symbol
- Productions \( P \) of the form
  - \( A \rightarrow Bf, \) push index \((f \in I)\)
  - \( Af \rightarrow B, \) pop index \((f \in I)\)
  - \( A \rightarrow uBv, \) generate terminals \((u, v \in T^*)\)
  - \( A \rightarrow BC, \) split and duplicate index word
  - \( A \rightarrow w, \) generate only terminals \((w \in T^*)\)

\[
S \rightarrow Sf, \quad S \rightarrow Sg, \quad S \rightarrow UU, \quad U \rightarrow \varepsilon, \\
Uf \rightarrow A, \quad Ug \rightarrow B, \quad A \rightarrow Ua, \quad B \rightarrow Ub.
\]

\( \mathcal{N} = \{S, T, A, B\}, I = \{f, g\}, T = \{a, b\}. \)
Application toIndexed Languages

No exact representation

Undecidable: Does $L \subseteq a^* b^*$ intersect with $\{a^n b^n \mid n \geq 0\}$?
Application to Indexed Languages

No exact representation

Undecidable: Does \( L \subseteq a^* b^* \) intersect with \( \{ a^n b^n \mid n \geq 0 \} ? \)

Given: indexed grammar \( G \) with \( L = L(G) \subseteq a_1^* \cdots a_n^* \), wlog \( L = L \downarrow \).
Application to Indexed Languages

No exact representation

Undecidable: Does $L \subseteq a^* b^*$ intersect with $\{a^n b^n \mid n \geq 0\}$?

Given: indexed grammar $G$ with $L = L(G) \subseteq a_1^* \cdots a_n^*$, wlog $L = L\downarrow$.

Observation

- Consider the derivations for $a_1^k \cdots a_n^k$, $k \geq 0$.  

Georg Zetzsche  (TU KL)  
Computing Downward Closures  
ICALP 2015
Application to Indexed Languages

No exact representation

Undecidable: Does \( L \subseteq a^* b^* \) intersect with \( \{a^n b^n \mid n \geq 0\} \)?

Given: indexed grammar \( G \) with \( L = L(G) \subseteq a_1^* \cdots a_n^* \), wlog \( L = L\downarrow \).

Observation

- Consider the derivations for \( a_1^k \cdots a_n^k \), \( k \geq 0 \).
- For each \( a_i \), at least one of the following holds:
Application to Indexed Languages

No exact representation

Undecidable: Does $L \subseteq a^* b^*$ intersect with $\{a^n b^n \mid n \geq 0\}$?

Given: indexed grammar $G$ with $L = L(G) \subseteq a_1^* \cdots a_n^*$, wlog $L = L\downarrow$.

Observation

- Consider the derivations for $a_1^k \cdots a_n^k$, $k \geq 0$.
- For each $a_i$, at least one of the following holds:
  - there is an unbounded number subtrees with yield in $a_i^*$
Application to Indexed Languages

No exact representation

Undecidable: Does $L \subseteq a^*b^*$ intersect with $\{a^n b^n \mid n \geq 0\}$?

Given: indexed grammar $G$ with $L = L(G) \subseteq a_1^* \cdots a_n^*$, wlog $L = L\downarrow$.

Observation

- Consider the derivations for $a_1^k \cdots a_n^k$, $k \geq 0$.
- For each $a_i$, at least one of the following holds:
  - there is an unbounded number subtrees with yield in $a_i^*$
  - the yields of such subtrees are unbounded in length
Application to Indexed Languages

No exact representation

Undecidable: Does $L \subseteq a^*b^*$ intersect with $\{a^n b^n \mid n \geq 0\}$?

Given: indexed grammar $G$ with $L = L(G) \subseteq a_1^* \cdots a_n^*$, wlog $L = L\downarrow$.

Observation

- Consider the derivations for $a_1^k \cdots a_n^k$, $k \geq 0$.
- For each $a_i$, at least one of the following holds:
  - there is an unbounded number subtrees with yield in $a_i^*$
  - the yields of such subtrees are unbounded in length

Step 1: Direct and indirect letters

For each subset $D \subseteq \{a_1, \ldots, a_n\}$, construct $G_D$
Application to Indexed Languages

No exact representation

Undecidable: Does $L \subseteq a^*b^*$ intersect with $\{a^n b^n \mid n \geq 0\}$?

Given: indexed grammar $G$ with $L = L(G) \subseteq a_1^* \cdots a_n^*$, wlog $L = L\downarrow$.

Observation

- Consider the derivations for $a_1^k \cdots a_n^k$, $k \geq 0$.
- For each $a_i$, at least one of the following holds:
  - there is an unbounded number subtrees with yield in $a_i^*$
  - the yields of such subtrees are unbounded in length

Step 1: Direct and indirect letters

For each subset $D \subseteq \{a_1, \ldots, a_n\}$, construct $G_D$:

- for $a_i \in D$, instead of deriving whole $a_i$-subtree, generate one $a_i$
- for $a_i \notin D$, derive only one of the $a_i$-subtrees
Application to Indexed Languages

No exact representation

Undecidable: Does $L \subseteq a^* b^*$ intersect with $\{a^n b^n \mid n \geq 0\}$?

Given: indexed grammar $G$ with $L = L(G) \subseteq a_1^* \cdots a_n^*$, wlog $L = L\downarrow$.

Observation

- Consider the derivations for $a_1^k \cdots a_n^k$, $k \geq 0$.
- For each $a_i$, at least one of the following holds:
  - there is an unbounded number of subtrees with yield in $a_i^*$
  - the yields of such subtrees are unbounded in length

Step 1: Direct and indirect letters

For each subset $D \subseteq \{a_1, \ldots, a_n\}$, construct $G_D$:

- for $a_i \in D$, instead of deriving whole $a_i$-subtree, generate one $a_i$
- for $a_i \notin D$, derive only one of the $a_i$-subtrees ← “indirect”
Application to Indexed Languages

No exact representation

Undecidable: Does \( L \subseteq a^* b^* \) intersect with \( \{a^n b^n \mid n \geq 0\} \)?

Given: indexed grammar \( G \) with \( L = L(G) \subseteq a_1^* \cdots a_n^* \), wlog \( L = L^\downarrow \).

Observation

- Consider the derivations for \( a_1^k \cdots a_n^k \), \( k \geq 0 \).
- For each \( a_i \), at least one of the following holds:
  - there is an unbounded number subtrees with yield in \( a_i^* \)
  - the yields of such subtrees are unbounded in length

Step 1: Direct and indirect letters

For each subset \( D \subseteq \{a_1, \ldots, a_n\} \), construct \( G_D \):

- for \( a_i \in D \), instead of deriving whole \( a_i \)-subtree, generate one \( a_i \)
- for \( a_i \notin D \), derive only one of the \( a_i \)-subtrees ← “indirect”

Then, \( a_1^* \cdots a_n^* \subseteq L(G)^\downarrow \) iff \( a_1^* \cdots a_n^* \subseteq L(G_D)^\downarrow \) for some \( D \).
Goal: bound nonterminal occurrences

Only obstacle: $a_i$-subtrees for indirect $a_i$
Goal: bound nonterminal occurrences

Only obstacle: $a_i$-subtrees for indirect $a_i$

- Consider the interval $a_i^* \cdots a_j^*$ for each occurring nonterminal
Goal: bound nonterminal occurrences

Only obstacle: $a_i$-subtrees for indirect $a_i$

- Consider the interval $a_i^* \cdots a_j^*$ for each occurring nonterminal

### Interval Consideration

Suppose: no unfolding of $a_i$-subtrees, indirect $a_i$

Then the nonterminals have pairwise distinct intervals

Therefore: Replace these subtrees with linear ones

### Symbolic Representation

\[ a_1 S_{(1,2)} a_2 a_2 T_{(3)} U_{(4)} a_5 V_{(5,8)} a_7 a_8 a_8 W_{(9)} \]

Indirect symbols: \{a_3, a_4, a_9\}
Goal: bound nonterminal occurrences

Only obstacle: $a_i$-subtrees for indirect $a_i$

- Consider the interval $a_i^* \cdots a_j^*$ for each occurring nonterminal
- Suppose: no unfolding of $a_i$-subtrees, indirect $a_i$

\[
a_1 S_{(1,2)} a_2 a_2 T_{(3)} U_{(4)} a_5 V_{(5,8)} a_7 a_8 a_8 W_{(9)}
\]

Indirect symbols: \{a_3, a_4, a_9\}
Goal: bound nonterminal occurrences

Only obstacle: $a_i$-subtrees for indirect $a_i$

- Consider the interval $a_i^* \cdots a_j^*$ for each occurring nonterminal
- Suppose: no unfolding of $a_i$-subtrees, indirect $a_i$
- Then the nonterminals have pairwise distinct intervals

$$a_1 S_{(1,2)} a_2 a_2 T_{(3)} U_{(4)} a_5 V_{(5,8)} a_7 a_8 a_8 W_{(9)}$$

Indirect symbols: $\{a_3, a_4, a_9\}$
Goal: bound nonterminal occurrences

Only obstacle: $a_i$-subtrees for indirect $a_i$

- Consider the interval $a_i^* \cdots a_j^*$ for each occurring nonterminal
- Suppose: no unfolding of $a_i$-subtrees, indirect $a_i$
- Then the nonterminals have pairwise distinct intervals

$\Rightarrow$ Bounded number of occurrences

\[
a_1 S_{(1,2)} a_2 a_2 T_{(3)} U_{(4)} a_5 V_{(5,8)} a_7 a_8 a_8 W_{(9)}
\]

Indirect symbols: $\{a_3, a_4, a_9\}$
Goal: bound nonterminal occurrences

Only obstacle: $a_i$-subtrees for indirect $a_i$

- Consider the interval $a_i^* \cdots a_j^*$ for each occurring nonterminal
- Suppose: no unfolding of $a_i$-subtrees, indirect $a_i$
- Then the nonterminals have pairwise distinct intervals

$\Rightarrow$ Bounded number of occurrences

Therefore: Replace these subtrees with linear ones

\[
a_1 S_{(1,2)} a_2 a_2 T_{(3)} U_{(4)} a_5 V_{(5,8)} a_7 a_8 a_8 W_{(9)}
\]

Indirect symbols: \{a_3, a_4, a_9\}
Goal: bound nonterminal occurrences

Only obstacle: $a_i$-subtrees for indirect $a_i$

- Consider the interval $a_i^* \cdots a_j^*$ for each occurring nonterminal
- Suppose: no unfolding of $a_i$-subtrees, indirect $a_i$
- Then the nonterminals have pairwise distinct intervals

$\Rightarrow$ Bounded number of occurrences

Therefore: Replace these subtrees with linear ones

$$a_1 S_{(1,2)} a_2 a_2 T_{(3)} U_{(4)} a_5 V_{(5,8)} a_7 a_8 a_8 W_{(9)}$$

Indirect symbols: $\{a_3, a_4, a_9\}$

Idea

Instead of unfolding $a_i$-subtree with root $Au$, $u \in I^*$, apply transducer to $u$
Goal: bound nonterminal occurrences

Only obstacle: \( a_i \)-subtrees for indirect \( a_i \)
- Consider the interval \( a_i^* \cdots a_j^* \) for each occurring nonterminal
- Suppose: no unfolding of \( a_i \)-subtrees, indirect \( a_i \)
- Then the nonterminals have pairwise distinct intervals
  \[ \rightarrow \text{Bounded number of occurrences} \]

Therefore: Replace these subtrees with linear ones

\[ a_1 S_{(1,2)} a_2 a_2 T_{(3)} U_{(4)} a_5 V_{(5,8)} a_7 a_8 a_8 W_{(9)} \]

Indirect symbols: \( \{ a_3, a_4, a_9 \} \)

Idea

Instead of unfolding \( a_i \)-subtree with root \( Au, u \in I^* \), apply transducer to \( u \)

However: Precise simulation not possible
Preserving $a_1^* \cdots a_n^* \subseteq L(G)\downarrow$

For transduction $T \subseteq NI^* \times a_i^*$, let $f_T, f_G : NI^* \rightarrow \mathbb{N} \cup \{\infty\}$ be

$$f_T(Au) = \sup\{|v| \mid (Au, v) \in T\}$$
$$f_G(Au) = \sup\{|v| \mid v \in a_i^*, \ Au \Rightarrow_G^* v\}$$
Preserving $a_1^* \cdots a_n^* \subseteq L(G)\downarrow$

For transduction $T \subseteq N^* \times a_i^*$, let $f_T, f_G: N^* \rightarrow \mathbb{N} \cup \{\infty\}$ be

$$f_T(Au) = \sup\{|v| \mid (Au, v) \in T\}$$

$$f_G(Au) = \sup\{|v| \mid v \in a_i^*, Au \Rightarrow_G^* v\}$$

Proposition

For each indexed grammar $G$, one can construct a rational transduction $T$ with $f_T \approx f_G$.

$f \approx g$: $f$ is unbounded on the same subsets as $g$  
(→ regular cost functions)
Preserving \( a_1^* \cdots a_n^* \subseteq L(G) \downarrow \)

For transduction \( T \subseteq N I^* \times a_i^* \), let \( f_T, f_G : N I^* \rightarrow \mathbb{N} \cup \{ \infty \} \) be

\[
f_T(Au) = \sup\{|v| \mid (Au, v) \in T\}
\]

\[
f_G(Au) = \sup\{|v| \mid v \in a_i^*, \ A u \Rightarrow_G^* v\}
\]

Proposition

For each indexed grammar \( G \), one can construct a rational transduction \( T \) with \( f_T \approx f_G \).

\( f \approx g \): \( f \) is unbounded on the same subsets as \( g \)

(\( \rightarrow \) regular cost functions)

Step 2: Apply transducer

- Only one nonterminal occurrence for transducer
Preserving $a_1^* \cdots a_n^* \subseteq L(G)$

For transduction $T \subseteq N^* \times a_i^*$, let $f_T, f_G : N^* \rightarrow \mathbb{N} \cup \{\infty\}$ be

$$f_T(Au) = \sup\{|v| \mid (Au, v) \in T\}$$
$$f_G(Au) = \sup\{|v| \mid v \in a_i^*, Au \Rightarrow^*_G v\}$$

Proposition

For each indexed grammar $G$, one can construct a rational transduction $T$ with $f_T \approx f_G$.

$f \approx g$: $f$ is unbounded on the same subsets as $g$

($\rightarrow$ regular cost functions)

Step 2: Apply transducer

- Only one nonterminal occurrence for transducer
  $\Rightarrow$ Bound on nonterminal occurrences, “breadth-bounded”
Remaining problem

- Given: Breadth-bounded indexed grammar $G$, $L(G) \subseteq a_1^* \cdots a_n^*$
- Is $a_1^* \cdots a_n^*$ included in $L(G)\downarrow$?
Remaining problem

- Given: Breadth-bounded indexed grammar \( G \), \( L(G) \subseteq a_1^* \cdots a_n^* \)
- Is \( a_1^* \cdots a_n^* \) included in \( L(G)^\downarrow \)?

Step 3:

Proposition

_Breadth-bounded indexed grammars have effectively semilinear Parikh images._
Thank you for your attention!