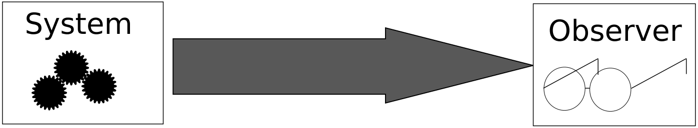
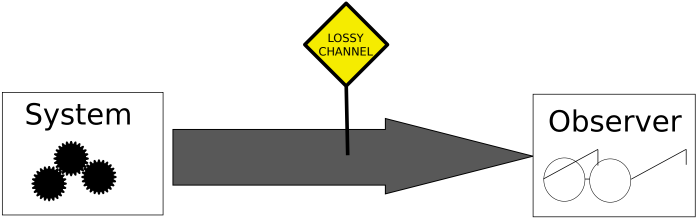
Downward Closures of Indexed Languages

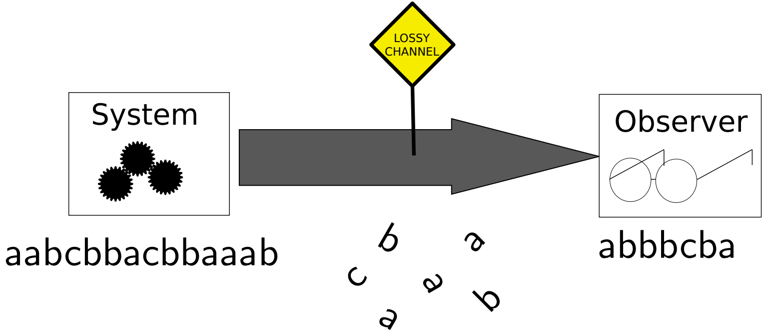
Georg Zetzsche

Technische Universität Kaiserslautern

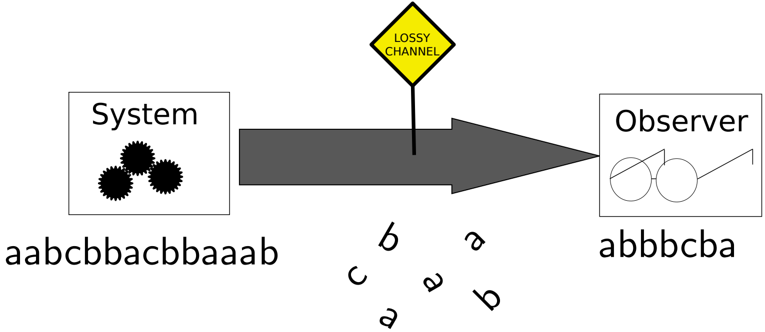
HOPA 2015


System
Observer


System
LOSSY
CHANNEL
Observer



$\underline{a}ab\underline{c}bb\underline{a}cb\underline{b}a\underline{aab}$



aabcbbacbbaaab

Downward Closures

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

Downward Closures

Theorem (Higman/Haines)

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

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? ($a^* \subseteq L{\downarrow}$)
- Is a ever executed after b? ($ab \in L{\downarrow}$)

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? ($ab \in L{\downarrow}$)
- 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?

Positive results

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 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 2001)

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

Context-free rules $A \to w$, 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 2001)

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

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

Theorem (Habermehl, Meyer, Wimmel 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 2001)

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

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

Theorem (Habermehl, Meyer, Wimmel 2010)

Downward closures are computable for Petri net languages.

Theorem (Z. 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 2001)

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

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

Theorem (Habermehl, Meyer, Wimmel 2010)

Downward closures are computable for Petri net languages.

Theorem (Z. 2015)

Downward closures are computable for stacked counter automata.

- Weak form of stack nesting
- Adding Counters

Negative results

Theorem (Gruber, Holzer, Kutrib 2009)

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

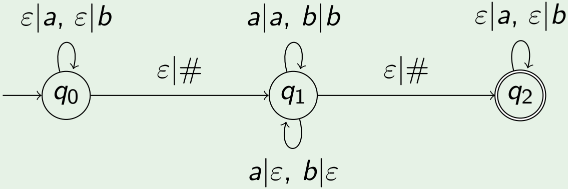
Theorem (Mayr 2003)

The reachability set of lossy channel systems is not computable.

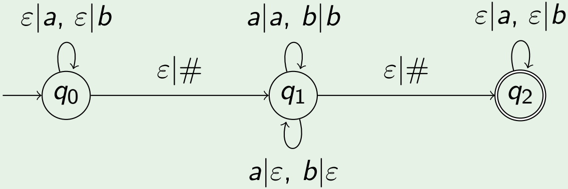
Theorem (Z. 2015)

Downward closures are computable for indexed languages, i.e. for second-order pushdown automata.

Example (Transducer)

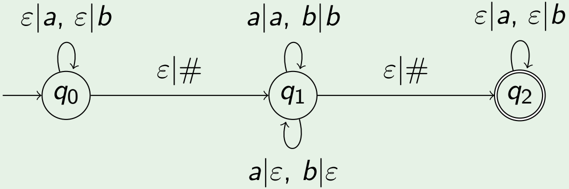


Example (Transducer)



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

Example (Transducer)



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

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

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

Fact (Aho 1968)

For every indexed language L and rational transduction T, the language TL is indexed as well.

Theorem (Z. 2015)

Let $\mathcal{C}$ be a language class that is closed under rational transductions. Then downward closures are computable for $\mathcal{C}$ if and only if the following problem is decidable:

Given	*A language $L \subseteq a_1^* \cdots a_n^*$ in $\mathcal{C}$*
Question	*Does $L{\downarrow}$ equal $a_1^* \cdots a_n^*$?*

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 $L\!\downarrow$ 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"

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

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

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

"Simple Regular Languages"

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.

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"

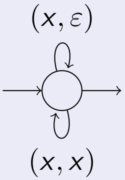
Algorithm

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.

Observation

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



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"

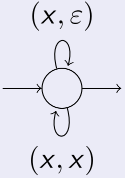
Algorithm

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

- $L{\downarrow} \subseteq R$ iff $L{\downarrow} \cap (X^*\setminus R) = \varnothing \rightsquigarrow$ 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 $\mathcal{C}$:



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 $(abc)^n$ for every $n \geqslant 0$.

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 $(abc)^n$ for every $n \geqslant 0$.

$$abc\ abc\ abc\ abc\ abc$$

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 $(abc)^n$ for every $n \geqslant 0$.

$$abc\ abc\ abc\ abc\ abc$$

$$bacca$$

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 $(abc)^n$ for every $n \geqslant 0$.

$$abc\ abc\ abc\ abc\ abc$$

$$bacca$$

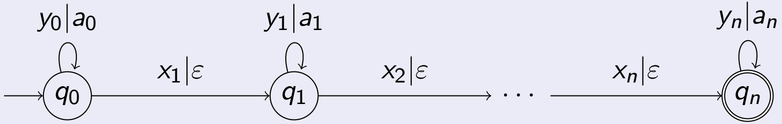
Observation

- 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 $(abc)^n$ for every $n \geqslant 0$.

$$abc\ abc\ abc\ abc\ abc$$

$$bacca$$

Transduction T



y_i: word containing each letter of Y_i once.

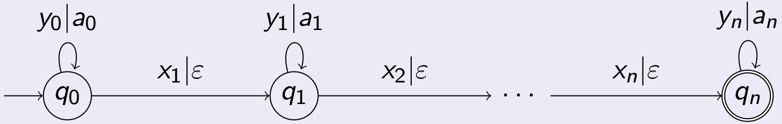
Observation

- 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 $(abc)^n$ for every $n \geqslant 0$.

$$abc\ abc\ abc\ abc\ abc$$

$$bacca$$

Transduction T



y_i: word containing each letter of Y_i once. Then:

$$T(L{\downarrow}){\downarrow} = a_0^*\cdots a_n^* \quad \text{iff} \quad Y_0^*\{x_1,\varepsilon\}Y_1^*\cdots\{x_n,\varepsilon\}Y_n^* \subseteq L{\downarrow}$$

Indexed Grammars

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
  - $A \to Bf$, push index ($f \in I$)
  - $Af \to B$, pop index ($f \in I$)
  - $A \to uBv$, generate terminals ($u, v \in T^*$)
  - $A \to BC$, split and duplicate index word
  - $A \to w$, 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
  - $A \to Bf$, push index ($f \in I$)
  - $Af \to B$, pop index ($f \in I$)
  - $A \to uBv$, generate terminals ($u, v \in T^*$)
  - $A \to BC$, split and duplicate index word
  - $A \to w$, generate only terminals ($w \in T^*$)

$$\begin{array}{llll} S \to Sf, & S \to Sg, & S \to UU, & U \to \varepsilon, \\ Uf \to A, & Ug \to B, & A \to Ua, & B \to Ub. \end{array}$$

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

Indexed Grammars

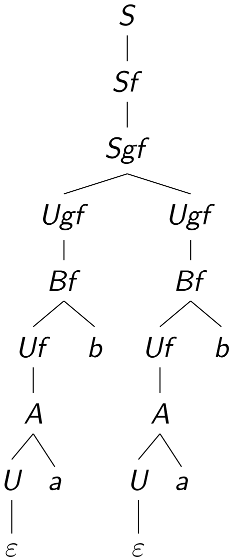
Indexed Grammars

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
  - $A \to Bf$, push index ($f \in I$)
  - $Af \to B$, pop index ($f \in I$)
  - $A \to uBv$, generate terminals ($u, v \in T^*$)
  - $A \to BC$, split and duplicate index word
  - $A \to w$, generate only terminals ($w \in T^*$)

$$S \to Sf, \quad S \to Sg, \quad S \to UU, \quad U \to \varepsilon,$$
$$Uf \to A, \quad Ug \to B, \quad A \to Ua, \quad B \to Ub.$$

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



Application to Indexed Languages

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

Observation

- Suppose $L\downarrow = a_1^* \cdots a_n^*$.
- Consider the derivations for $a_1^k \cdots a_n^k$, $k \geqslant 0$.

Application to Indexed Languages

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

Observation

- Suppose $L\downarrow = a_1^* \cdots a_n^*$.
- Consider the derivations for $a_1^k \cdots a_n^k$, $k \geqslant 0$.
- For each a_i, at least one of the following holds:

Application to Indexed Languages

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

Observation

- Suppose $L{\downarrow} = a_1^* \cdots a_n^*$.
- Consider the derivations for $a_1^k \cdots a_n^k$, $k \geqslant 0$.
- 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

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

Observation

- Suppose $L\downarrow = a_1^* \cdots a_n^*$.
- Consider the derivations for $a_1^k \cdots a_n^k$, $k \geqslant 0$.
- 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

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

Observation

- Suppose $L\!\downarrow = a_1^* \cdots a_n^*$.
- Consider the derivations for $a_1^k \cdots a_n^k$, $k \geqslant 0$.
- 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

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

Observation

- Suppose $L{\downarrow} = a_1^* \cdots a_n^*$.
- Consider the derivations for $a_1^k \cdots a_n^k$, $k \geqslant 0$.
- 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

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

Observation

- Suppose $L\!\downarrow = a_1^* \cdots a_n^*$.
- Consider the derivations for $a_1^k \cdots a_n^k$, $k \geqslant 0$.
- 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.

Then, $L(G)\!\downarrow = a_1^* \cdots a_n^*$ iff $L(G_D)\!\downarrow = a_1^* \cdots a_n^*$ for some D.

Goal: bound nonterminal occurrences

Only obstacle: a_i-subtrees for $a_i \notin D$

Goal: bound nonterminal occurrences

Only obstacle: a_i-subtrees for $a_i \notin D$

- Suppose we did not unfold them

Goal: bound nonterminal occurrences

Only obstacle: a_i-subtrees for $a_i \notin D$

- Suppose we did not unfold them
- Consider the interval $a_i^* \cdots a_j^*$ for each occurring nonterminal

Goal: bound nonterminal occurrences

Only obstacle: a_i-subtrees for $a_i \notin D$

- Suppose we did not unfold them
- Consider the interval $a_i^* \cdots a_j^*$ for each occurring nonterminal
- Then the nonterminals have pairwise distinct intervals

Goal: bound nonterminal occurrences

Only obstacle: a_i-subtrees for $a_i \notin D$

- Suppose we did not unfold them
- Consider the interval $a_i^* \cdots a_j^*$ for each occurring nonterminal
- Then the nonterminals have pairwise distinct intervals

$\Rightarrow$ Bounded number of occurrences

Goal: bound nonterminal occurrences

Only obstacle: a_i-subtrees for $a_i \notin D$

- Suppose we did not unfold them
- Consider the interval $a_i^* \cdots a_j^*$ for each occurring nonterminal
- Then the nonterminals have pairwise distinct intervals

$\Rightarrow$ Bounded number of occurrences

Therefore: Replace these subtrees with linear ones

Goal: bound nonterminal occurrences

Only obstacle: a_i-subtrees for $a_i \notin D$

- Suppose we did not unfold them
- Consider the interval $a_i^* \cdots a_j^*$ for each occurring nonterminal
- Then the nonterminals have pairwise distinct intervals

$\Rightarrow$ Bounded number of occurrences

Therefore: Replace these subtrees with linear ones

Idea

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

Preserving $L(G)\downarrow = a_1^* \cdots a_n^*$

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

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

Preserving $L(G)\downarrow = a_1^* \cdots a_n^*$

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

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

Proposition

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

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

Preserving $L(G)\!\downarrow = a_1^* \cdots a_n^*$

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

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

Proposition

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

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

Step 2: Apply transducer

- Instead of unfolding a_i-subtrees, $a_i \notin D$, apply transducer to index word.

Preserving $L(G)\!\downarrow = a_1^* \cdots a_n^*$

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

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

Proposition

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

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

Step 2: Apply transducer

- Instead of unfolding a_i-subtrees, $a_i \notin D$, apply transducer to index word.
- Only one nonterminal occurrence for transducer

Preserving $L(G){\downarrow} = a_1^* \cdots a_n^*$

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

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

Proposition

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

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

Step 2: Apply transducer

- Instead of unfolding a_i-subtrees, $a_i \notin D$, apply transducer to index word.
- Only one nonterminal occurrence for transducer

$\Rightarrow$ Bound on nonterminal occurrences, "breadth-bounded"

Remaining problem

- Given: Breadth-bounded indexed grammar G, $L(G) \subseteq a_1^* \cdots a_n^*$
- Does $L(G){\downarrow} = a_1^* \cdots a_n^*$?

Remaining problem

- Given: Breadth-bounded indexed grammar G, $L(G) \subseteq a_1^* \cdots a_n^*$
- Does $L(G)\downarrow = a_1^* \cdots a_n^*$?

Step 3:

Proposition

Breadth-bounded indexed grammars have effectively semilinear Parikh images.

Remaining problem

- Given: Breadth-bounded indexed grammar G, $L(G) \subseteq a_1^* \cdots a_n^*$
- Does $L(G){\downarrow} = a_1^* \cdots a_n^*$?

Step 3:

Proposition

Breadth-bounded indexed grammars have effectively semilinear Parikh images.

Then, it is clearly decidable whether $L(G){\downarrow} = a_1^* \cdots a_n^*$.

Thank you for your attention!