Finite Automata for the Sub- and Superword Closure of CFLs: Descriptional and Computational Complexity

Maximilian Schlund

joint work with
Georg Bachmeier and Michael Luttenberger

Department of Computer Science
TU München

March 4, 2015

Subwords and Closure

$$w \preccurlyeq u :\Leftrightarrow \; w \text{ subword (subsequence) of } u$$

$$aabcab \preccurlyeq bbcababaacbcbabacc$$

Subwords and Closure

$$w \preccurlyeq u :\Leftrightarrow \; w \text{ subword (subsequence) of } u$$

$$aabcab \preccurlyeq bbcababaacbcbabacc$$

Def.: Subword Closure of a Language

$$\nabla L := \{w \in \Sigma^* : \exists u \in L. w \preccurlyeq u\}$$

Example

$$L = \{a^n c b^n : n \in \mathbb{N}\} \;\rightsquigarrow\; \nabla L = a^*(\varepsilon + c)b^*$$

Subword Closure: Computation

Higman/Haines, 1950s

∇L is regular for any language L.

Subword Closure: Computation

Higman/Haines, 1950s

∇L is regular for any language L.

But beware. . .

$$\nabla L = \emptyset \Leftrightarrow L = \emptyset$$

$\rightsquigarrow$ Automaton for ∇L not computable in general (e.g. for context-sensitive languages).

Subword Closure: Computation

Higman/Haines, 1950s

∇L is regular for any language L.

But beware...

$$\nabla L = \emptyset \Leftrightarrow L = \emptyset$$

$\rightsquigarrow$ Automaton for ∇L not computable in general (e.g. for context-sensitive languages).

Effective regularity for CFLs (van Leeuwen '78, Courcelle '91)

∇L computable for context-free languages (as automaton/regex).

Subword Closure: Computation

Higman/Haines, 1950s

∇L is regular for any language L.

But beware...

$$\nabla L = \emptyset \Leftrightarrow L = \emptyset$$

$\leadsto$ Automaton for ∇L not computable in general (e.g. for context-sensitive languages).

Effective regularity for CFLs (van Leeuwen '78, Courcelle '91)

∇L computable for context-free languages (as automaton/regex).

Question: Size of finite automata representing ∇L (DFA/NFA)?

Size of NFA for the Subword Closure of a CFL

Context-free grammar G, NFA $\mathcal{A}^{\nabla}$ for $\nabla L(G)$,

- Gruber/Holzer/Kutrib '09: $\left|\mathcal{A}^{\nabla}\right| \in 2^{2^{\mathcal{O}(|G|)}}$, based on [vL78].

Size of NFA for the Subword Closure of a CFL

Context-free grammar G, NFA $\mathcal{A}^\nabla$ for $\nabla L(G)$,

- Gruber/Holzer/Kutrib '09: $\left|\mathcal{A}^\nabla\right| \in 2^{2^{\mathcal{O}(|G|)}}$, based on [vL78].
- We show $\left|\mathcal{A}^\nabla\right| \in 2^{\mathcal{O}(|G|)}$, based on [Cou91].

Size of NFA for the Subword Closure of a CFL

Context-free grammar G, NFA $\mathcal{A}^\nabla$ for $\nabla L(G)$,

- Gruber/Holzer/Kutrib '09: $\left|\mathcal{A}^\nabla\right| \in 2^{2^{\mathcal{O}(|G|)}}$, based on [vL78].
- We show $\left|\mathcal{A}^\nabla\right| \in 2^{\mathcal{O}(|G|)}$, based on [Cou91].

Known lower bound $\left|\mathcal{A}^\nabla\right| \in \Omega(2^{|G|})$:

Consider $L_n = \{a^{2^n}\}$, context-free grammar with $|G| \in \mathcal{O}(n)$:

$$\begin{aligned} A_n &\to A_{n-1}A_{n-1} \\ &\vdots \\ A_1 &\to A_0A_0 \\ A_0 &\to a \end{aligned}$$

NFAs for L_n and $\nabla L_n = \{a^i : 0 \leq i \leq 2^n\}$ need 2^n states.

Grammar $\rightarrow$ NFA, pre-processing

1. Assume G in 2-normal form: $X \rightarrow \alpha$, $|\alpha| \leq 2$ (w.l.o.g.).
2. If $X \Rightarrow^* \alpha X \beta X \gamma$ then $\nabla X = \Sigma_X^*$ $\rightsquigarrow$ non-expansive grammar.

Grammar $\to$ NFA, pre-processing

1. Assume G in 2-normal form: $X \to \alpha$, $|\alpha| \leq 2$ (w.l.o.g.).
2. If $X \Rightarrow^* \alpha X \beta X \gamma$ then $\nabla X = \Sigma_X^*$ $\rightsquigarrow$ non-expansive grammar.

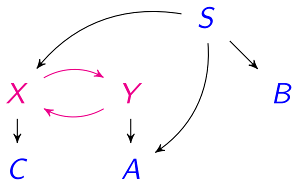
$$
\begin{aligned}
S &\to XA \mid AB \mid AX \mid XB \\
X &\to CY \mid C \\
Y &\to XA \mid A \\
A &\to a \\
B &\to b \\
C &\to c
\end{aligned}
$$

$$
\begin{tikzcd}
 & & S \arrow[dll, bend right] \arrow[dl, bend left] \arrow[dr] & \\
X \arrow[r, bend left] \arrow[d] & Y \arrow[l, bend left] \arrow[d] & & B \\
C & A & &
\end{tikzcd}
$$

Grammar $\to$ NFA, pre-processing

1. Assume G in 2-normal form: $X \to \alpha$, $|\alpha| \leq 2$ (w.l.o.g.).
2. If $X \Rightarrow^* \alpha X \beta X \gamma$ then $\nabla X = \Sigma_X^*$ $\rightsquigarrow$ non-expansive grammar.
3. Contract strongly-connected components.

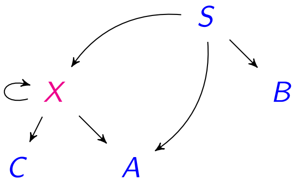
$$
\begin{aligned}
S &\to XA \mid AB \mid AX \mid XB \\
X &\to CY \mid C \\
Y &\to XA \mid A \\
A &\to a \\
B &\to b \\
C &\to c
\end{aligned}
$$



Grammar $\to$ NFA, pre-processing

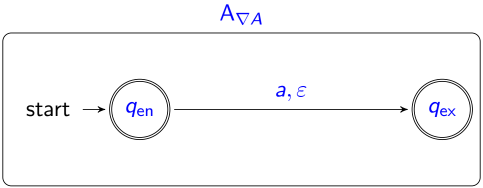
1. Assume G in 2-normal form: $X \to \alpha$, $|\alpha| \leq 2$ (w.l.o.g.).
2. If $X \Rightarrow^* \alpha X \beta X \gamma$ then $\nabla X = \Sigma_X^*$ $\rightsquigarrow$ non-expansive grammar.
3. Contract strongly-connected components.
4. Dependency graph is now a DAG (with self-loops)

$$
\begin{aligned}
S &\to XA \mid AB \mid AX \mid XB \\
X &\to CX \mid XA \mid C \mid A \\
A &\to a \\
B &\to b \\
C &\to c
\end{aligned}
$$



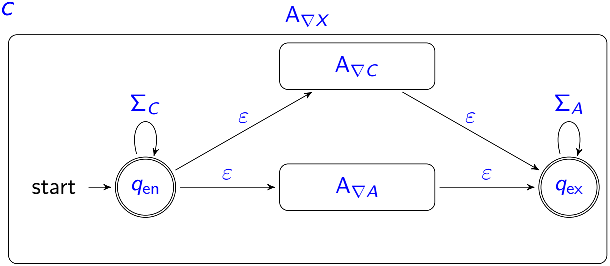
Bottom-Up Construction and Re-Use of Automata

$$
\begin{aligned}
S &\rightarrow XA \mid AB \mid AX \mid XB \\
X &\rightarrow CX \mid XA \mid C \mid A \\
A &\rightarrow a \\
B &\rightarrow b \\
C &\rightarrow c
\end{aligned}
$$



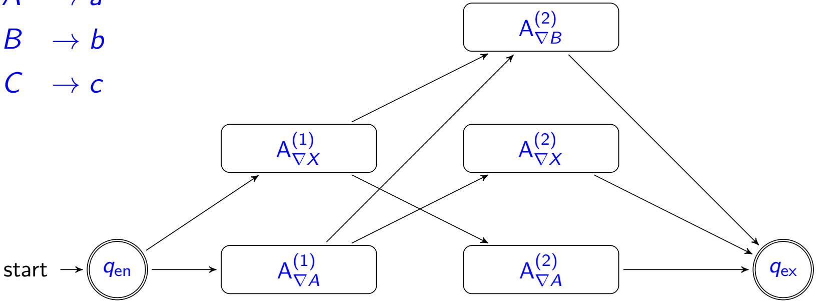
Bottom-Up Construction and Re-Use of Automata

$$
\begin{aligned}
S &\to XA \mid AB \mid AX \mid XB \\
X &\to CX \mid XA \mid C \mid A \\
A &\to a \\
B &\to b \\
C &\to c
\end{aligned}
$$



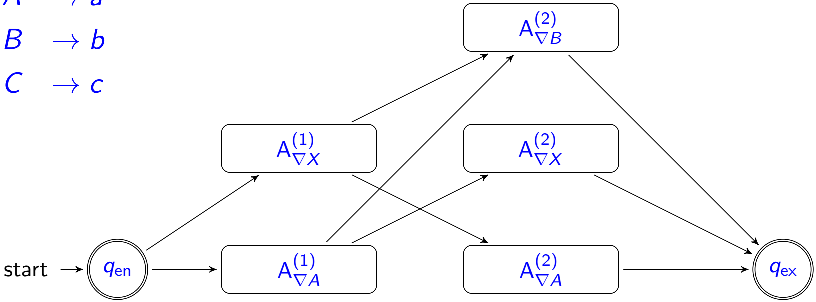
Bottom-Up Construction and Re-Use of Automata

$$S \rightarrow XA \mid AB \mid AX \mid XB$$
$$X \rightarrow CX \mid XA \mid C \mid A$$
$$A \rightarrow a$$
$$B \rightarrow b$$
$$C \rightarrow c$$



Bottom-Up Construction and Re-Use of Automata

$$
\begin{aligned}
S &\to XA \mid AB \mid AX \mid XB \\
X &\to CX \mid XA \mid C \mid A \\
A &\to a \\
B &\to b \\
C &\to c
\end{aligned}
$$



Total size of NFA

At every stage, copy each automaton at most twice
$\rightsquigarrow \left|\mathcal{A}^{\nabla}\right| \in 2^{\mathcal{O}(|G|)}$.

Size of DFA for the Subword Closure of a CFL L

- NFA for ∇L: $\left|\mathcal{A}^{\nabla}\right| \in 2^{\mathcal{O}(|G|)} \rightsquigarrow$ DFA: $\left|\mathcal{D}^{\nabla}\right| \in 2^{2^{\mathcal{O}(|G|)}}$.

Size of DFA for the Subword Closure of a CFL L

- NFA for ∇L: $\left|\mathcal{A}^{\nabla}\right| \in 2^{\mathcal{O}(|G|)} \rightsquigarrow$ DFA: $\left|\mathcal{D}^{\nabla}\right| \in 2^{2^{\mathcal{O}(|G|)}}$.
- Bound is tight: $w \in \{0,1\}^{2N+1}$, $w = x0y0z$ with $|y| = N$ (finite language, binary alphabet).

$$L_N := \bigcup_{j=1}^{N} \underbrace{\{0,1\}^{j-1}}_{x}\{0\}\underbrace{\{0,1\}^{N}}_{y}\{0\}\underbrace{\{0,1\}^{N-j}}_{z}.$$

Size of DFA for the Subword Closure of a CFL L

- NFA for ∇L: $\left|\mathcal{A}^{\nabla}\right| \in 2^{\mathcal{O}(|G|)} \rightsquigarrow$ DFA: $\left|\mathcal{D}^{\nabla}\right| \in 2^{2^{\mathcal{O}(|G|)}}$.
- Bound is tight: $w \in \{0,1\}^{2N+1}$, $w = x0y0z$ with $|y| = N$ (finite language, binary alphabet).

$$L_N := \bigcup_{j=1}^{N} \underbrace{\{0,1\}^{j-1}}_{x}\{0\}\underbrace{\{0,1\}^{N}}_{y}\{0\}\underbrace{\{0,1\}^{N-j}}_{z}.$$

- $|G| \in \mathcal{O}(\log N)$: let derivation "guess" the location of y.
- $|\mathcal{D}| \geq 2^N$: all $u \in \{0,1\}^N$ inequivalent w.r.t. Myhill-Nerode.
- $\left|\mathcal{D}^{\nabla}\right| \geq 2^N$ (use same witnesses for inequivalence).

Application: Approximate Grammar Equivalence Checking

Given two context-free languages $L_1 = L(G_1), L_2 = L(G_2)$,

$$\nabla L_1 \neq \nabla L_2 \implies L_1 \neq L_2$$

Application: Approximate Grammar Equivalence Checking

Given two context-free languages $L_1 = L(G_1), L_2 = L(G_2)$,

$$\nabla L_1 \neq \nabla L_2 \implies L_1 \neq L_2$$

Questions:

- Complexity of checking $\nabla L_1 \stackrel{?}{=} \nabla L_2$ (given as NFAs)?
- How to build a semi-decision procedure?
  - If $\nabla L_1 \neq \nabla L_2$ output witness $w \in (L_1 \setminus L_2) \cup (L_2 \setminus L_1)$.
  - If $\nabla L_1 = \nabla L_2$ refine grammars and iterate.

Equivalence of NFAs modulo Closure

For NFAs $\mathcal{A}_1, \mathcal{A}_2$,

$$\mathcal{A}_1 \equiv_{\mathsf{cl}} \mathcal{A}_2 \text{ if } \mathsf{cl}(L_1) = \mathsf{cl}(L_2).$$

- $\mathcal{A}_1 \stackrel{?}{\equiv} \mathcal{A}_2$: PSPACE-complete.

Equivalence of NFAs modulo Closure

For NFAs $\mathcal{A}_1, \mathcal{A}_2$,

$$\mathcal{A}_1 \equiv_{\mathsf{cl}} \mathcal{A}_2 \text{ if } \mathsf{cl}(L_1) = \mathsf{cl}(L_2).$$

- $\mathcal{A}_1 \stackrel{?}{\equiv} \mathcal{A}_2$: PSPACE-complete.
- $\mathcal{A}_1 \stackrel{?}{\equiv}_{\mathsf{cl}} \mathcal{A}_2$: stays PSPACE-complete for $\mathsf{cl} =$prefix/suffix/factor closure (Rampersad/Shallit/Xu 2012).

Equivalence of NFAs modulo Closure

For NFAs $\mathcal{A}_1, \mathcal{A}_2$,

$$\mathcal{A}_1 \equiv_{\mathsf{cl}} \mathcal{A}_2 \text{ if } \mathsf{cl}(L_1) = \mathsf{cl}(L_2).$$

- $\mathcal{A}_1 \stackrel{?}{\equiv} \mathcal{A}_2$: PSPACE-complete.
- $\mathcal{A}_1 \stackrel{?}{\equiv}_{\mathsf{cl}} \mathcal{A}_2$: stays PSPACE-complete for $\mathsf{cl} =$prefix/suffix/factor closure (Rampersad/Shallit/Xu 2012).
- Our result: $\mathcal{A}_1 \stackrel{?}{\equiv}_{\nabla} \mathcal{A}_2$ is (only) coNP-complete.

(I) $\mathcal{A}_1 \overset{?}{\equiv}_{\nabla} \mathcal{A}_2$ is in coNP

Structure of $\mathcal{A}^{\nabla}$ and its powerset-DFA:

- If $s \xrightarrow{a} t$ then $s \xrightarrow{\varepsilon} t$ in $\mathcal{A}^{\nabla}$.
- If $S \xrightarrow{a} T$ in powerset-DFA, then $S \supseteq T$.

(I) $\mathcal{A}_1 \stackrel{?}{\equiv}_\nabla \mathcal{A}_2$ is in coNP

Structure of $\mathcal{A}^\nabla$ and its powerset-DFA:

- If $s \xrightarrow{a} t$ then $s \xrightarrow{\varepsilon} t$ in $\mathcal{A}^\nabla$.
- If $S \xrightarrow{a} T$ in powerset-DFA, then $S \supseteq T$.

$\Longrightarrow$ Longest acyclic path in powerset-DFA has length $\leq |\mathcal{A}^\nabla|$.

(I) $\mathcal{A}_1 \stackrel{?}{\equiv}_\nabla \mathcal{A}_2$ is in coNP

Structure of $\mathcal{A}^\nabla$ and its powerset-DFA:

- If $s \xrightarrow{a} t$ then $s \xrightarrow{\varepsilon} t$ in $\mathcal{A}^\nabla$.
- If $S \xrightarrow{a} T$ in powerset-DFA, then $S \supseteq T$.

$\Longrightarrow$ Longest acyclic path in powerset-DFA has length $\leq |\mathcal{A}^\nabla|$.

Corollary: In-equivalence modulo ∇ is in NP

If $\mathcal{A}_1 \not\equiv_\nabla \mathcal{A}_2$ then there is a short witness w ($|w| \leq |\mathcal{A}_1^\nabla| + |\mathcal{A}_2^\nabla|$).

(II) $\mathcal{A}_1 \overset{?}{\equiv}_{\nabla} \mathcal{A}_2$ is coNP-hard

Even more: $\mathcal{A}_1 \overset{?}{\equiv}_{\nabla} \mathcal{A}_2$ is coNP-hard for finite languages.

Same idea as hardness of equivalence for regex without Kleene-stars.

(II) $\mathcal{A}_1 \stackrel{?}{\equiv}_\nabla \mathcal{A}_2$ is coNP-hard

Even more: $\mathcal{A}_1 \stackrel{?}{\equiv}_\nabla \mathcal{A}_2$ is coNP-hard for finite languages.

Same idea as hardness of equivalence for regex without Kleene-stars.

Standard reduction from TAUT:

- Given: Propositional formula φ (in DNF) with n variables.

(II) $\mathcal{A}_1 \stackrel{?}{\equiv}_\nabla \mathcal{A}_2$ is coNP-hard

Even more: $\mathcal{A}_1 \stackrel{?}{\equiv}_\nabla \mathcal{A}_2$ is coNP-hard for finite languages.

Same idea as hardness of equivalence for regex without Kleene-stars.

Standard reduction from TAUT:

- Given: Propositional formula φ (in DNF) with n variables.
- Build regex ρ, $L(\rho) \in \{0,1\}^n$ that enumerates all satisfying assignments of φ.

(II) $\mathcal{A}_1 \stackrel{?}{\equiv}_\nabla \mathcal{A}_2$ is coNP-hard

Even more: $\mathcal{A}_1 \stackrel{?}{\equiv}_\nabla \mathcal{A}_2$ is coNP-hard for finite languages.

Same idea as hardness of equivalence for regex without Kleene-stars.

Standard reduction from TAUT:

- Given: Propositional formula φ (in DNF) with n variables.
- Build regex ρ, $L(\rho) \in \{0,1\}^n$ that enumerates all satisfying assignments of φ.
- $\varphi \in \mathrm{TAUT} \iff L(\rho) = \{0,1\}^n \iff \nabla L(\rho) = \{0,1\}^{\leq n}$.

(II) $\mathcal{A}_1 \stackrel{?}{\equiv}_\nabla \mathcal{A}_2$ is coNP-hard

Even more: $\mathcal{A}_1 \stackrel{?}{\equiv}_\nabla \mathcal{A}_2$ is coNP-hard for finite languages.

Same idea as hardness of equivalence for regex without Kleene-stars.

Standard reduction from TAUT:

- Given: Propositional formula φ (in DNF) with n variables.
- Build regex ρ, $L(\rho) \in \{0,1\}^n$ that enumerates all satisfying assignments of φ.
- $\varphi \in \text{TAUT} \iff L(\rho) = \{0,1\}^n \iff \nabla L(\rho) = \{0,1\}^{\leq n}$.
- Last step: $L(\rho) \neq \{0,1\}^n \implies \nabla L(\rho) \neq \{0,1\}^{\leq n}$ since ∇ only adds shorter words.

Summary, Future Work

- Tight bounds on the size of NFAs/DFAs for $\nabla L(G)$.
- $\mathcal{A}_1 \overset{?}{\equiv}_\nabla \mathcal{A}_2$ is (only) coNP-complete.
- All results hold for superword closures as well (see paper).
- Application: Semi-decision procedure for grammar inequivalence (fast in practice – see paper).

Summary, Future Work

- Tight bounds on the size of NFAs/DFAs for $\nabla L(G)$.
- $\mathcal{A}_1 \stackrel{?}{\equiv}_\nabla \mathcal{A}_2$ is (only) coNP-complete.
- All results hold for superword closures as well (see paper).
- Application: Semi-decision procedure for grammar inequivalence (fast in practice – see paper).

Current/Future Work:

- Simple regex as representation (size $2^{\mathcal{O}(|G|)}$ as well), equivalence in $\mathcal{O}(n^2)$ (Abdulla/Bouajjani/Jonsson '98).
- Develop practical tool for inclusion/equivalence checking.
- Computability of ∇L for larger language classes.

Summary, Future Work

- Tight bounds on the size of NFAs/DFAs for $\nabla L(G)$.
- $\mathcal{A}_1 \overset{?}{\equiv}_\nabla \mathcal{A}_2$ is (only) coNP-complete.
- All results hold for superword closures as well (see paper).
- Application: Semi-decision procedure for grammar inequivalence (fast in practice – see paper).

Current/Future Work:

- Simple regex as representation (size $2^{\mathcal{O}(|G|)}$ as well), equivalence in $\mathcal{O}(n^2)$ (Abdulla/Bouajjani/Jonsson '98).
- Develop practical tool for inclusion/equivalence checking.
- Computability of ∇L for larger language classes.

Thank you!