INF2080

Context-Sensitive Langugaes

Daniel Lupp

Universitetet i Oslo

15th February 2017

Department of Informatics

University of Oslo

Context-Free Grammar

Definition (Context-Free Grammar)

A *context-free grammar* is a 4-tuple (V, Σ, R, S) where

1. V is a finite set of *variables*
2. Σ is a finite set disjoint from V of *terminals*
3. R is a finite set of *rules*, each consisting of a variable and of a string of variables and terminals
4. and S is the *start variable*

Rules are of the form $A \rightarrow B_1 B_2 B_3 \ldots B_m$, where $A \in V$ and each $B_i \in V \cup \Sigma$.

Why Context-Sensitive?

- Many building blocks of programming languages are context-free, *but not all!*

Why Context-Sensitive?

- Many building blocks of programming languages are context-free, *but not all!*
- consider the following toy programming language, where you can “declare” and “assign” a variable a value.

$$S \rightarrow \texttt{declare}\ \ v; S \mid \texttt{assign}\ \ v : x; S$$

Why Context-Sensitive?

- Many building blocks of programming languages are context-free, *but not all!*
- consider the following toy programming language, where you can "declare" and "assign" a variable a value.

$$S \rightarrow \texttt{declare}\ v; S \mid \texttt{assign}\ v : x; S$$

- this is context-free...

Why Context-Sensitive?

- Many building blocks of programming languages are context-free, *but not all!*
- consider the following toy programming language, where you can "declare" and "assign" a variable a value.

$$S \rightarrow \texttt{declare}\ \ v; S \mid \texttt{assign}\ \ v : x; S$$

- this is context-free...
- but what if we only want to allow assignment after declaration and an infinite amount of variable names? $\rightarrow$ context-sensitive!

Context-Sensitive Languages

- Some believe natural languages reside in the class of context-sensitive languages, though this is a controversial topic amongst linguists.

Context-Sensitive Languages

- Some believe natural languages reside in the class of context-sensitive languages, though this is a controversial topic amongst linguists.
- But many characteristics of natural languages (e.g., verb-noun agreement) are context-sensitive!

Context-Sensitive Grammars

So, instead of allowing for a single variable on the left-hand side of a rule, we allow for a *context*:

$$\alpha B\gamma \to \alpha\beta\gamma \tag{1}$$

with $\alpha, \beta, \gamma \in (V \cup \Sigma)^*$, but $\beta \neq \varepsilon$.

Context-Sensitive Grammars

So, instead of allowing for a single variable on the left-hand side of a rule, we allow for a *context*:

$$\alpha B \gamma \to \alpha \beta \gamma \tag{1}$$

with $\alpha, \beta, \gamma \in (V \cup \Sigma)^*$, but $\beta \neq \varepsilon$.

Definition (Context-sensitive grammar)

A *context-sensitive grammar* is a 4-tuple (V, Σ, R, S) consisting of

- a finite set V of *variables*;
- a finite set Σ of *terminals*, disjoint from V;
- a set R of *rules* of the form (1);
- a start variable $S \in V$. If S does not occur on any righthand side of a rule in S, we also allow for the rule $S \to \varepsilon$ in R.

Example

Recall that the language $\{a^n b^n c^n \mid n \geq 1\}$ was not context-free. (pumping lemma for CFLs)

Example

Recall that the language $\{a^n b^n c^n \mid n \geq 1\}$ was not context-free. (pumping lemma for CFLs)
It is, however, context-sensitive!

Example

Recall that the language $\{a^n b^n c^n \mid n \geq 1\}$ was not context-free. (pumping lemma for CFLs)
It is, however, context-sensitive!
A context-sensitive grammar that produces this language:

$$
\begin{aligned}
S &\to ABC \\
S &\to ASB'C \\
CB' &\to Z_1B' \\
Z_1B' &\to Z_1Z_2 \\
Z_1Z_2 &\to B'Z_2
\end{aligned}
\qquad\qquad
\begin{aligned}
B'Z_2 &\to B'C \\
BB' &\to BB \\
A &\to a \\
B &\to b \\
C &\to c
\end{aligned}
$$

Noncontracting Grammars

So CSGs can be quite cumbersome...

Noncontracting Grammars

So CSGs can be quite cumbersome...many rules needed to encode, e.g., the swapping rule $cB \to Bc$.

Noncontracting Grammars

So CSGs can be quite cumbersome...many rules needed to encode, e.g., the swapping rule $cB \to Bc$.

Definition (Noncontracting Grammars)

A *noncontracting grammar* is a set of rules $\alpha \to \beta$, where $\alpha, \beta \in (V \cup \Sigma)^*$ and $|\alpha| \leq |\beta|$. In addition, it may contain $S \to \varepsilon$ if S does not occur on any righthand side of a rule.

Noncontracting Grammars

So CSGs can be quite cumbersome...many rules needed to encode, e.g., the swapping rule $cB \to Bc$.

Definition (Noncontracting Grammars)

A *noncontracting grammar* is a set of rules $\alpha \to \beta$, where $\alpha, \beta \in (V \cup \Sigma)^*$ and $|\alpha| \leq |\beta|$. In addition, it may contain $S \to \varepsilon$ if S does not occur on any righthand side of a rule.

Examples:

$$cC \to abABc$$
$$ab \to de$$
$$cB \to Bc$$

Noncontracting Grammars

So CSGs can be quite cumbersome...many rules needed to encode, e.g., the swapping rule $cB \to Bc$.

Definition (Noncontracting Grammars)

A *noncontracting grammar* is a set of rules $\alpha \to \beta$, where $\alpha, \beta \in (V \cup \Sigma)^*$ and $|\alpha| \leq |\beta|$. In addition, it may contain $S \to \varepsilon$ if S does not occur on any righthand side of a rule.

Examples:

$$cC \to abABc$$
$$ab \to de$$
$$cB \to Bc$$

Note: none of these rules are context-sensitive!

Noncontracting vs. Context-sensitive Grammars

- First note: context-sensitive rules $\alpha B\gamma \to \alpha\beta\gamma$ are noncontracting, since we required $\beta \neq \varepsilon$.

Noncontracting vs. Context-sensitive Grammars

- First note: context-sensitive rules $\alpha B\gamma \to \alpha\beta\gamma$ are noncontracting, since we required $\beta \neq \varepsilon$.
- So it would seem that noncontracting grammars are quite a bit more expressive than context-sensitive grammars. After all, $\alpha \to \beta$ is far more general...

Noncontracting vs. Context-sensitive Grammars

- First note: context-sensitive rules $\alpha B\gamma \to \alpha\beta\gamma$ are noncontracting, since we required $\beta \neq \varepsilon$.
- So it would seem that noncontracting grammars are quite a bit more expressive than context-sensitive grammars. After all, $\alpha \to \beta$ is far more general...

Theorem

Every noncontracting grammar can be transformed into a context-sensitive grammar that produces the same language.

Noncontracting vs. Context-sensitive Grammars

- First note: context-sensitive rules $\alpha B\gamma \to \alpha\beta\gamma$ are noncontracting, since we required $\beta \neq \varepsilon$.
- So it would seem that noncontracting grammars are quite a bit more expressive than context-sensitive grammars. After all, $\alpha \to \beta$ is far more general...

Theorem

Every noncontracting grammar can be transformed into a context-sensitive grammar that produces the same language.

So, in the spirit of INF2080's love of abbreviations: NCG = CSG!

Example

The language $\{a^n b^n c^n \mid n \geq 1\}$ described by CSG:

$$
\begin{aligned}
S &\rightarrow ABC & B'Z_2 &\rightarrow B'C \\
S &\rightarrow ASB'C & BB' &\rightarrow BB \\
CB' &\rightarrow Z_1B' & A &\rightarrow a \\
Z_1B' &\rightarrow Z_1Z_2 & B &\rightarrow b \\
Z_1Z_2 &\rightarrow B'Z_2 & C &\rightarrow c
\end{aligned}
$$

Example

The language $\{a^n b^n c^n \mid n \geq 1\}$ described by NCG:

$$
\begin{aligned}
S &\rightarrow abc \\
S &\rightarrow aSBc \\
cB &\rightarrow Bc \\
bB &\rightarrow bb
\end{aligned}
$$

Example

The language $\{a^n b^n c^n \mid n \geq 1\}$ described by NCG:

$$\begin{aligned} S &\rightarrow abc \\ S &\rightarrow aSBc \\ cB &\rightarrow Bc \\ bB &\rightarrow bb \end{aligned}$$

Due to the equivalence, some people define context-senstive languages using noncontracting grammars.

Kuroda Normal Form

Similar to CFG's Chomsky Normal Form, CSG's have a normal form of their own:

Definition (Kuroda Normal Form)

A noncontracting grammar is in *Kuroda normal form* if all rules are of the form

$$\begin{aligned} AB &\rightarrow CD \\ A &\rightarrow BC \\ A &\rightarrow B \\ A &\rightarrow a \end{aligned}$$

for variables A, B, C, D and terminals a.

Kuroda Normal Form

Similar to CFG's Chomsky Normal Form, CSG's have a normal form of their own:

Definition (Kuroda Normal Form)

A noncontracting grammar is in *Kuroda normal form* if all rules are of the form

$$
\begin{aligned}
AB &\rightarrow CD \\
A &\rightarrow BC \\
A &\rightarrow B \\
A &\rightarrow a
\end{aligned}
$$

for variables A, B, C, D and terminals a.

Theorem

For every context-sensitive grammar there exists a noncontracting grammar in Kuroda normal form that produces the same language.

Linear Bounded Automata

So what type of computational model accepts precisely the context-sensitive languages?

Linear Bounded Automata

So what type of computational model accepts precisely the context-sensitive languages?
$\rightarrow$ linear bounded automata!

Linear Bounded Automata

So what type of computational model accepts precisely the context-sensitive languages?
$\rightarrow$ linear bounded automata!
Essentially, LBAs are NFAs with a bounded tape as memory (as opposed to the stack for PDAs)

Linear Bounded Automata

So what type of computational model accepts precisely the context-sensitive languages?
$\rightarrow$ linear bounded automata!
Essentially, LBAs are NFAs with a bounded tape as memory (as opposed to the stack for PDAs)

Definition

A *linear bounded automaton* (LBA) is a tuple $(Q, \Sigma, \Gamma, \delta, <, >, q_0, q_a, q_r)$ where $Q, \Sigma, \Gamma, \delta, q_0, q_a, q_r$ are defined precisely as in a Turing machine, except that the transition function can neither move the head to the left of the *left marker* $<$ nor to the right of the *right marker* $>$.

Linear Bounded Automata

So what type of computational model accepts precisely the context-sensitive languages?
$\rightarrow$ linear bounded automata!
Essentially, LBAs are NFAs with a bounded tape as memory (as opposed to the stack for PDAs)

Definition

A *linear bounded automaton* (LBA) is a tuple $(Q, \Sigma, \Gamma, \delta, <, >, q_0, q_a, q_r)$ where $Q, \Sigma, \Gamma, \delta, q_0, q_a, q_r$ are defined precisely as in a Turing machine, except that the transition function can neither move the head to the left of the *left marker* $<$ nor to the right of the *right marker* $>$.

A LBA initializes in the configuration $< q_0 w_1 w_2 \cdots w_n >$. So, intuitively, the tape of the Turing machine is restricted to the length of the input.

Myhill-Landweber-Kuroda Theorem

Theorem (Myhill-Landweber-Kuroda Theorem)

A language is context-sensitive iff there exists a linear bounded automaton that recognizes it.

Myhill-Landweber-Kuroda Theorem

Theorem (Myhill-Landweber-Kuroda Theorem)

A language is context-sensitive iff there exists a linear bounded automaton that recognizes it.

- Proof sketch for "$\Rightarrow$": Construct an automaton with a tape containing the input, nondeterministically guesses a rule in the languages grammar and applies the rule "backwards", i.e., replaces the input's symbols occuring on the righthand side (RHS) of the rule with the lefthand side (LHS) of the rule.

Myhill-Landweber-Kuroda Theorem

Theorem (Myhill-Landweber-Kuroda Theorem)

A language is context-sensitive iff there exists a linear bounded automaton that recognizes it.

- Proof sketch for "$\Rightarrow$": Construct an automaton with a tape containing the input, nondeterministically guesses a rule in the languages grammar and applies the rule "backwards", i.e., replaces the input's symbols occuring on the righthand side (RHS) of the rule with the lefthand side (LHS) of the rule.
- Example: Let $aabbcc$ be the input and let the grammar contain the rule $bB \rightarrow bb$. Then applying this rule "backwards" on $aabbcc$ yields the string $aabBcc$.

Myhill-Landweber-Kuroda Theorem

Theorem (Myhill-Landweber-Kuroda Theorem)

A language is context-sensitive iff there exists a linear bounded automaton that recognizes it.

- Proof sketch for "$\Rightarrow$": Construct an automaton with a tape containing the input, nondeterministically guesses a rule in the languages grammar and applies the rule "backwards", i.e., replaces the input's symbols occuring on the righthand side (RHS) of the rule with the lefthand side (LHS) of the rule.
- Example: Let $aabbcc$ be the input and let the grammar contain the rule $bB \rightarrow bb$. Then applying this rule "backwards" on $aabbcc$ yields the string $aabBcc$.
- If the input can thus be rewritten to the start variable S, the machine accepts.

Myhill-Landweber-Kuroda Theorem

Theorem (Myhill-Landweber-Kuroda Theorem)

A language is context-sensitive iff there exists a linear bounded automaton that recognizes it.

- Proof sketch for "$\Rightarrow$": Construct an automaton with a tape containing the input, nondeterministically guesses a rule in the languages grammar and applies the rule "backwards", i.e., replaces the input's symbols occuring on the righthand side (RHS) of the rule with the lefthand side (LHS) of the rule.
- Example: Let $aabbcc$ be the input and let the grammar contain the rule $bB \rightarrow bb$. Then applying this rule "backwards" on $aabbcc$ yields the string $aabBcc$.
- If the input can thus be rewritten to the start variable S, the machine accepts. Since the grammar is noncontracting, i.e. $|\text{LHS}| \leq |\text{RHS}|$, the string cannot get longer through backwards application of rules. thus the head of the machine never has to leave the scope of the input.

Myhill-Landweber-Kuroda Theorem

Theorem (Myhill-Landweber-Kuroda Theorem)

A language is context-sensitive iff there exists a linear bounded automaton that recognizes it.

- Proof sketch for "$\Rightarrow$": Construct an automaton with a tape containing the input, nondeterministically guesses a rule in the languages grammar and applies the rule "backwards", i.e., replaces the input's symbols occuring on the righthand side (RHS) of the rule with the lefthand side (LHS) of the rule.
- Example: Let $aabbcc$ be the input and let the grammar contain the rule $bB \rightarrow bb$. Then applying this rule "backwards" on $aabbcc$ yields the string $aabBcc$.
- If the input can thus be rewritten to the start variable S, the machine accepts. Since the grammar is noncontracting, i.e. $|\text{LHS}| \leq |\text{RHS}|$, the string cannot get longer through backwards application of rules. thus the head of the machine never has to leave the scope of the input.$\rightarrow$ this machine is a LBA.

Myhill-Landweber-Kuroda Theorem

Theorem (Myhill-Landweber-Kuroda Theorem)

A language is context-sensitive iff there exists a linear bounded automaton that recognizes it.

- Proof sketch for "$\Rightarrow$": Construct an automaton with a tape containing the input, nondeterministically guesses a rule in the languages grammar and applies the rule "backwards", i.e., replaces the input's symbols occuring on the righthand side (RHS) of the rule with the lefthand side (LHS) of the rule.
- Example: Let $aabbcc$ be the input and let the grammar contain the rule $bB \rightarrow bb$. Then applying this rule "backwards" on $aabbcc$ yields the string $aabBcc$.
- If the input can thus be rewritten to the start variable S, the machine accepts. Since the grammar is noncontracting, i.e. $|\text{LHS}| \leq |\text{RHS}|$, the string cannot get longer through backwards application of rules. thus the head of the machine never has to leave the scope of the input.$\rightarrow$ this machine is a LBA.
- Proof for "$\Leftarrow$" much more involved.

Closure properties of CSLs

- Union $L_1 \cup L_2$

Closure properties of CSLs

- Union $L_1 \cup L_2$: add new start variable $\overline{S}$ and rule $\overline{S} \rightarrow S_1 | S_2$.
- Concatanation $L_1 L_2$

Closure properties of CSLs

- Union $L_1 \cup L_2$: add new start variable $\overline{S}$ and rule $\overline{S} \rightarrow S_1 | S_2$.
- Concatanation $L_1 L_2$: add new start variable $\overline{S}$ and rule $\overline{S} \rightarrow S_1 S_2$.
- Kleene star L_1^*

Closure properties of CSLs

- Union $L_1 \cup L_2$: add new start variable $\overline{S}$ and rule $\overline{S} \rightarrow S_1 | S_2$.
- Concatanation $L_1 L_2$: add new start variable $\overline{S}$ and rule $\overline{S} \rightarrow S_1 S_2$.
- Kleene star L_1^*: add new start variable $\overline{S}$ and rules $\overline{S} \rightarrow \varepsilon | S_1 S_1$
- Reversal $L_1^R = \{w^R \mid w \in L_1\}$

Closure properties of CSLs

- Union $L_1 \cup L_2$: add new start variable $\overline{S}$ and rule $\overline{S} \to S_1 | S_2$.
- Concatanation $L_1 L_2$: add new start variable $\overline{S}$ and rule $\overline{S} \to S_1 S_2$.
- Kleene star L_1^*: add new start variable $\overline{S}$ and rules $\overline{S} \to \varepsilon | S_1 S_1$
- Reversal $L_1^R = \{w^R \mid w \in L_1\}$: create grammar that contains a rule $\gamma^R B \alpha^R \to \gamma^R \beta^R \alpha^R$ for each rule $\alpha B \gamma \to \alpha \beta \gamma$ in the grammar of L_1.
- Intersection $L_1 \cap L_2$

Closure properties of CSLs

- Union $L_1 \cup L_2$: add new start variable $\overline{S}$ and rule $\overline{S} \to S_1|S_2$.
- Concatanation $L_1 L_2$: add new start variable $\overline{S}$ and rule $\overline{S} \to S_1 S_2$.
- Kleene star L_1^*: add new start variable $\overline{S}$ and rules $\overline{S} \to \varepsilon|S_1 S_1$
- Reversal $L_1^R = \{w^R \mid w \in L_1\}$: create grammar that contains a rule $\gamma^R B \alpha^R \to \gamma^R \beta^R \alpha^R$ for each rule $\alpha B \gamma \to \alpha \beta \gamma$ in the grammar of L_1.
- Intersection $L_1 \cap L_2$: Use multitape LBAs (equivalent to LBA, without proof). Simulate the computation for each language on a separate tape; if both accept, the automaton accepts.
- Recall that context-free languages are *not* closed under intersection and complementation!

LBA Problems

But what about the complement of a context-sensitive language?

LBA Problems

But what about the complement of a context-sensitive language? Kuroda phrased two open problems related to LBAs in the 60's.

LBA Problems

But what about the complement of a context-sensitive language? Kuroda phrased two open problems related to LBAs in the 60's.

- **First LBA Problem:** Are nondeterministic LBA's equivalent to deterministic LBA's?
- equivalent complexity theoretic question: is $\text{NSPACE}(O(n)) = \text{DSPACE}(O(n))$?

LBA Problems

But what about the complement of a context-sensitive language? Kuroda phrased two open problems related to LBAs in the 60's.

- **First LBA Problem:** Are nondeterministic LBA's equivalent to deterministic LBA's?
- equivalent complexity theoretic question: is NSPACE($O(n)$) = DSPACE($O(n)$)?
- **Second LBA Problem:** Is the class of languages accepted by LBA's closed under complementation?
- equivalent complexity theoretic question: is NSPACE($O(n)$) = co-NSPACE($O(n)$)?

LBA Problems

But what about the complement of a context-sensitive language? Kuroda phrased two open problems related to LBAs in the 60's.

- **First LBA Problem:** Are nondeterministic LBA's equivalent to deterministic LBA's?
- equivalent complexity theoretic question: is $\text{NSPACE}(O(n)) = \text{DSPACE}(O(n))$?
- **Second LBA Problem:** Is the class of languages accepted by LBA's closed under complementation?
- equivalent complexity theoretic question: is $\text{NSPACE}(O(n)) = \text{co-NSPACE}(O(n))$?

The first problem is still an open question, while the second was answered in 1988 by Immerman and Szelepscényi.

Complement of CSLs

Theorem (Immerman-Szelepcsényi Theorem)

$\mathrm{NSPACE}(O(n)) = \text{co-}\mathrm{NSPACE}(O(n))$.

And hence

Theorem

The class of context-sensitive languages is closed under complementation.

Decidability spoilers for the next weeks

We will soon have a look at some decidability results of the various classes of languages we have seen:

	$x \in L$	$L = \emptyset$	$L = \Sigma^*$	$L = K$	$L \cap K = \emptyset$
regular	✓	✓	✓	✓	✓
(DCFL	✓	✓	✓	✓	X)
CFL	✓	✓	X	X	X
CSL	✓	X	X	X	X
decidable	✓	X	X	X	X
Turing-rec.	X	X	X	X	X

Chomsky Hierarchy

- Type-0: recursively enumerable, i.e., Turing-recognizable languages.
- Type-1: context-sensitive languages.
- Type-2: context-free languages.
- Type-3: regular languages.

Chomsky Hierarchy

- Type-0: recursively enumerable, i.e., Turing-recognizable languages.
- Type-1: context-sensitive languages.
- Type-2: context-free languages.
- Type-3: regular languages.

So we've seen/will see:
{Regular Languages} $\subsetneq$ {CFLs}$\subsetneq$ {CSLs} $\subsetneq$ {Turing-rec. Languages} and
{Regular Languages} $\subsetneq$ {CFLs}$\subsetneq$ {Decidable Languages} $\subsetneq$ {Turing-rec. Languages}.

Chomsky Hierarchy

- Type-0: recursively enumerable, i.e., Turing-recognizable languages.
- Type-1: context-sensitive languages.
- Type-2: context-free languages.
- Type-3: regular languages.

So we've seen/will see:
{Regular Languages} $\subsetneq$ {CFLs}$\subsetneq$ {CSLs} $\subsetneq$ {Turing-rec. Languages} and
{Regular Languages} $\subsetneq$ {CFLs}$\subsetneq$ {Decidable Languages} $\subsetneq$ {Turing-rec. Languages}.
But what is the relationship between {CSLs} and {Decidable Languages}?

Decidable vs. Context-sensitive

Without proof:

Theorem

The class of context-sensitive languages is decidable.

Decidable vs. Context-sensitive

Without proof:

Theorem

The class of context-sensitive languages is decidable.

Hence, $\{\text{CSLs}\} \subseteq \{\text{Decidable Languages}\}$.

Decidable vs. Context-sensitive

Without proof:

Theorem

The class of context-sensitive languages is decidable.

Hence, {CSLs} $\subseteq$ {Decidable Languages}. However, there exists a decidable language that is *not* context-sensitive!

Decidable vs. Context-sensitive

Without proof:

Theorem

The class of context-sensitive languages is decidable.

Hence, {CSLs} $\subseteq$ {Decidable Languages}. However, there exists a decidable language that is *not* context-sensitive!

Let $L = \{w \mid \text{w is a string representation of a CSG } G \text{ and } w \notin L(G)\}$.

Decidable vs. Context-sensitive

Without proof:

Theorem

The class of context-sensitive languages is decidable.

Hence, {CSLs} $\subseteq$ {Decidable Languages}. However, there exists a decidable language that is *not* context-sensitive!

Let $L = \{w \mid \text{w is a string representation of a CSG } G \text{ and } w \notin L(G)\}$. First of all:

Theorem

L is decidable.

Decidable vs. Context-sensitive

Without proof:

Theorem

The class of context-sensitive languages is decidable.

Hence, {CSLs} $\subseteq$ {Decidable Languages}. However, there exists a decidable language that is *not* context-sensitive!

Let $L = \{w \mid \text{w is a string representation of a CSG } G \text{ and } w \notin L(G)\}$. First of all:

Theorem

L is decidable.

Proof idea: Given an input w, check if it represents a CSG. Then use the decider from the previous theorem to check whether $w \notin L(G)$.

A decidable, non-context-sensitive language

Let $L = \{w \mid \text{w is a string representation of a CSG } G \text{ and } w \notin L(G)\}$.

Theorem

L is not context-sensitive.

Proof idea: Assume L is context-sensitive. Then let w be a string representation of its CSG G.

A decidable, non-context-sensitive language

Let $L = \{w \mid \text{w is a string representation of a CSG } G \text{ and } w \notin L(G)\}$.

Theorem

L is not context-sensitive.

Proof idea: Assume L is context-sensitive. Then let w be a string representation of its CSG G. Question: Is $w \in L(G) = L$?

A decidable, non-context-sensitive language

Let $L = \{w \mid \text{w is a string representation of a CSG } G \text{ and } w \notin L(G)\}$.

Theorem

L is not context-sensitive.

Proof idea: Assume L is context-sensitive. Then let w be a string representation of its CSG G. Question: Is $w \in L(G) = L$?

- Assume $w \in L$. Then by definition of L, w is not contained in $L(G) = L$, i.e., $w \notin L$. Contradiction!

A decidable, non-context-sensitive language

Let $L = \{w \mid \text{w is a string representation of a CSG } G \text{ and } w \notin L(G)\}$.

Theorem

L is not context-sensitive.

Proof idea: Assume L is context-sensitive. Then let w be a string representation of its CSG G. Question: Is $w \in L(G) = L$?

- Assume $w \in L$. Then by definition of L, w is not contained in $L(G) = L$, i.e., $w \notin L$. Contradiction!
- Assume $w \notin L$. Then w represents a CSG and is not a member of of the language it represents. Hence, $w \in L(G) = L$. Contradiction!

A decidable, non-context-sensitive language

Let $L = \{w \mid \text{w is a string representation of a CSG } G \text{ and } w \notin L(G)\}$.

Theorem

L is not context-sensitive.

Proof idea: Assume L is context-sensitive. Then let w be a string representation of its CSG G. Question: Is $w \in L(G) = L$?

- Assume $w \in L$. Then by definition of L, w is not contained in $L(G) = L$, i.e., $w \notin L$. Contradiction!
- Assume $w \notin L$. Then w represents a CSG and is not a member of of the language it represents. Hence, $w \in L(G) = L$. Contradiction! □

$\Rightarrow$ {CSLs} $\subsetneq$ {Decidable Languages}!