Automata Theory :: Context-Sensitive Grammars and Linear Bounded Automata

Jörg Endrullis

Vrije Universiteit Amsterdam

Context-Sensitive Grammars

A grammar is **context-sensitive** if for every rule $x \to y$ it holds:

$$|x| \leq |y| \quad \text{(and } x \neq \lambda\text{)}$$

Note: the words cannot get shorter during derivation.

Context-Sensitive Grammars

A grammar is **context-sensitive** if for every rule $x \rightarrow y$ it holds:

$$|x| \leq |y| \quad \text{(and } x \neq \lambda\text{)}$$

Note: the words cannot get shorter during derivation.

For every context-sensitive grammar G_1
there exists a grammar G_2 with rules of the form

$$xAy \rightarrow xvy \quad \text{with} \quad v \neq \lambda$$

such that $L(G_1) = L(G_2)$.

(Compare with the shape of rules in a context-free grammar.)

Context-Sensitive Grammars

A grammar is **context-sensitive** if for every rule $x \to y$ it holds:

$$|x| \leq |y| \quad \text{(and } x \neq \lambda\text{)}$$

Note: the words cannot get shorter during derivation.

For every context-sensitive grammar G_1
there exists a grammar G_2 with rules of the form

$$xAy \to xvy \quad \text{with} \quad v \neq \lambda$$

such that $L(G_1) = L(G_2)$.

(Compare with the shape of rules in a context-free grammar.)

A language L is **context-sensitive** if there exists a
context-sensitive grammar G with $L(G) = L \setminus \{\lambda\}$.

Example

The language

$$\{\, a^n b^n c^n \mid n \geq 1 \,\}$$

is generated by the context-sensitive grammar:

$$\begin{aligned} S &\to aAbc \mid abc \\ A &\to aAB \mid aB \\ Bb &\to bB \\ Bc &\to bcc \end{aligned}$$

Example derivation:

$$\begin{aligned} S &\Rightarrow aAbc \Rightarrow aaABbc \Rightarrow aaAbBc \\ &\Rightarrow aaAbbcc \Rightarrow aaaBbbcc \Rightarrow aaabBbcc \\ &\Rightarrow aaabbBcc \Rightarrow aaabbbccc \end{aligned}$$

Linear Bounded Automata

Linear Bounded Automata

A **linear bounded automaton**, short **LBA**,
is a nondeterministic TM $(Q, \Sigma, \Gamma, \delta, q_0, F)$.

Note that there is no $\Box$!

Instead, we have symbols [and], and

- [and] are placed around the input word
- for every $q \in Q$, $\delta(q, [\,)$ is of the form $(q', [\,, R)$
- for every $q \in Q$, $\delta(q,]\,)$ is of the form $(q',]\,, L)$

The head can only move within the bounds of the input word!

Linear Bounded Automata

A **linear bounded automaton**, short **LBA**,
is a nondeterministic TM $(Q, \Sigma, \Gamma, \delta, q_0, F)$.

Note that there is no $\Box$!

Instead, we have symbols [and], and

- [and] are placed around the input word
- for every $q \in Q$, $\delta(q, [\,)$ is of the form $(q', [\,, R)$
- for every $q \in Q$, $\delta(q, \,])$ is of the form $(q', \,], L)$

The head can only move within the bounds of the input word!

So the memory is restricted by the length of the input word.

Linear Bounded Automata

A **linear bounded automaton**, short **LBA**,
is a nondeterministic TM $(Q, \Sigma, \Gamma, \delta, q_0, F)$.

Note that there is no $\square$!

Instead, we have symbols [and], and

- [and] are placed around the input word
- for every $q \in Q$, $\delta(q, [\,)$ is of the form $(q', [\,, R)$
- for every $q \in Q$, $\delta(q,]\,)$ is of the form $(q',]\,, L)$

The head can only move within the bounds of the input word!

So the memory is restricted by the length of the input word.

The **language** $L(M)$ **accepted by** LBA $M = (Q, \Sigma, \Gamma, \delta, q_0, F)$ is

$$\{\, w \in \Sigma^+ \mid q_0[w] \vdash^+ [uqv] \;\text{ for some } q \in F, u, v \in \Gamma^* \,\}$$

From Context-Sensitive Grammars to LBA's

Theorem

For every context-sensitive grammar G there exists an LBA M such that $L(M) = L(G)$.

From Context-Sensitive Grammars to LBA's

Theorem

For every context-sensitive grammar G there exists an LBA M such that $L(M) = L(G)$.

Proof.

A derivation of $w \in L(G)$ contains only words of length $\leq |w|$.

A nondeterministic Turing machine can simulate (guess) this derivation without leaving the bounds of w. $\square$

From LBA's to Context-Sensitive Grammars

Theorem

For every LBA M, the language $L(M)$ is context-sensitive.

From LBA's to Context-Sensitive Grammars

Theorem

For every LBA M, the language $L(M)$ is context-sensitive.

Proof sketch.

As before, build an unrestricted grammar G with $L(G) = L(M)$.

From LBA's to Context-Sensitive Grammars

Theorem

For every LBA M, the language $L(M)$ is context-sensitive.

Proof sketch.

As before, build an unrestricted grammar G with $L(G) = L(M)$.

All productions rules are context-sensitive, except for:

$$\square \rightarrow \lambda$$

From LBA's to Context-Sensitive Grammars

Theorem

For every LBA M, the language $L(M)$ is context-sensitive.

Proof sketch.

As before, build an unrestricted grammar G with $L(G) = L(M)$.

All productions rules are context-sensitive, except for:

$$\square \to \lambda$$

However, a linear bounded automaton does not use $\square$!
(It never leaves the borders of the input word.)

From LBA's to Context-Sensitive Grammars

Theorem

For every LBA M, the language $L(M)$ is context-sensitive.

Proof sketch.

As before, build an unrestricted grammar G with $L(G) = L(M)$.

All productions rules are context-sensitive, except for:

$$\Box \rightarrow \lambda$$

However, a linear bounded automaton does not use $\Box$!
(It never leaves the borders of the input word.)

Therefore, we can drop

- the rule $\Box \rightarrow \lambda$

From LBA's to Context-Sensitive Grammars

Theorem

For every LBA M, the language $L(M)$ is context-sensitive.

Proof sketch.

As before, build an unrestricted grammar G with $L(G) = L(M)$.

All productions rules are context-sensitive, except for:

$$\square \rightarrow \lambda$$

However, a linear bounded automaton does not use $\square$!
(It never leaves the borders of the input word.)

Therefore, we can drop

- the rule $\square \rightarrow \lambda$, and
- the rules $S \rightarrow V_{\square}^{\square} S \mid S V_{\square}^{\square}$.

From LBA's to Context-Sensitive Grammars

Theorem

For every LBA M, the language $L(M)$ is context-sensitive.

Proof sketch.

As before, build an unrestricted grammar G with $L(G) = L(M)$.

All productions rules are context-sensitive, except for:

$$\Box \to \lambda$$

However, a linear bounded automaton does not use $\Box$!
(It never leaves the borders of the input word.)

Therefore, we can drop

- the rule $\Box \to \lambda$, and
- the rules $S \to V_{\Box}^{\Box} S \mid S V_{\Box}^{\Box}$.

(In step 1, we derive from S a word $V_{q_0[a_1}^{a_1} V_{a_2}^{a_2} \cdots V_{a_{n-1}}^{a_{n-1}} V_{a_n]}^{a_n}$.)

Basic Properties of Context-Sensitive Languages

Basic Properties of Context-Sensitive Languages

Theorem

If L_1 and L_2 are context-sensitive, then so are

$$L_1 \cup L_2 \qquad L_1 \cap L_2 \qquad L_1^R \qquad L_1 L_2 \qquad L_1^* \qquad \overline{L_1} \qquad L_1 \setminus L_2$$

Basic Properties of Context-Sensitive Languages

Theorem

If L_1 and L_2 are context-sensitive, then so are

$$L_1 \cup L_2 \qquad L_1 \cap L_2 \qquad L_1^R \qquad L_1 L_2 \qquad L_1^* \qquad \overline{L_1} \qquad L_1 \setminus L_2$$

Proof.

Basic Properties of Context-Sensitive Languages

Theorem

If L_1 and L_2 are context-sensitive, then so are

$$L_1 \cup L_2 \qquad L_1 \cap L_2 \qquad L_1^R \qquad L_1 L_2 \qquad L_1^* \qquad \overline{L_1} \qquad L_1 \setminus L_2$$

Proof.

- $L_1 \cup L_2$, L_1^R, $L_1 L_2$: proof via grammars (same as before)

Basic Properties of Context-Sensitive Languages

Theorem

If L_1 and L_2 are context-sensitive, then so are

$$L_1 \cup L_2 \qquad L_1 \cap L_2 \qquad L_1^R \qquad L_1 L_2 \qquad L_1^* \qquad \overline{L_1} \qquad L_1 \setminus L_2$$

Proof.

- $L_1 \cup L_2$, L_1^R, $L_1 L_2$: proof via grammars (same as before)
- L_1^*: $S \rightarrow S_1 S \mid S_1$ where S is the fresh starting variable

Basic Properties of Context-Sensitive Languages

Theorem

If L_1 and L_2 are context-sensitive, then so are

$$L_1 \cup L_2 \qquad L_1 \cap L_2 \qquad L_1^R \qquad L_1 L_2 \qquad L_1^* \qquad \overline{L_1} \qquad L_1 \setminus L_2$$

Proof.

- $L_1 \cup L_2$, L_1^R, $L_1 L_2$: proof via grammars (same as before)
- L_1^*: $S \rightarrow S_1 S \mid S_1$ where S is the fresh starting variable
- $L_1 \cap L_2$: run both linear bounded automata in sequence

Basic Properties of Context-Sensitive Languages

Theorem

If L_1 and L_2 are context-sensitive, then so are

$$L_1 \cup L_2 \qquad L_1 \cap L_2 \qquad L_1^R \qquad L_1 L_2 \qquad L_1^* \qquad \overline{L_1} \qquad L_1 \setminus L_2$$

Proof.

- $L_1 \cup L_2$, L_1^R, $L_1 L_2$: proof via grammars (same as before)
- L_1^*: $S \to S_1 S \mid S_1$ where S is the fresh starting variable
- $L_1 \cap L_2$: run both linear bounded automata in sequence
- $L_1 \setminus L_2 = L_1 \cap \overline{L_2}$

Basic Properties of Context-Sensitive Languages

Theorem

If L_1 and L_2 are context-sensitive, then so are

$$L_1 \cup L_2 \qquad L_1 \cap L_2 \qquad L_1^R \qquad L_1 L_2 \qquad L_1^* \qquad \overline{L_1} \qquad L_1 \setminus L_2$$

Proof.

- $L_1 \cup L_2$, L_1^R, $L_1 L_2$: proof via grammars (same as before)
- L_1^*: $S \to S_1 S \mid S_1$ where S is the fresh starting variable
- $L_1 \cap L_2$: run both linear bounded automata in sequence
- $L_1 \setminus L_2 = L_1 \cap \overline{L_2}$
- $\overline{L_1}$: proven by Immerman and Szelepcsényi (1987) □

Basic Properties of Context-Sensitive Languages

Theorem

If L_1 and L_2 are context-sensitive, then so are

$$L_1 \cup L_2 \qquad L_1 \cap L_2 \qquad L_1^R \qquad L_1 L_2 \qquad L_1^* \qquad \overline{L_1} \qquad L_1 \setminus L_2$$

Proof.

- $L_1 \cup L_2$, L_1^R, $L_1 L_2$: proof via grammars (same as before)
- L_1^*: $S \to S_1 S \mid S_1$ where S is the fresh starting variable
- $L_1 \cap L_2$: run both linear bounded automata in sequence
- $L_1 \setminus L_2 = L_1 \cap \overline{L_2}$
- $\overline{L_1}$: proven by Immerman and Szelepcsényi (1987) □

It is unknown whether **deterministic** LBA's are equally expressive as **nondeterministic** LBA's.

Context-Sensitive vs. Recursive Languages

Context-Sensitive Languages are Recursive

Theorem

Context-sensitive languages are recursive.

Context-Sensitive Languages are Recursive

Theorem

Context-sensitive languages are recursive.

Proof.

Let G be a context-sensitive grammar.

Context-Sensitive Languages are Recursive

Theorem

Context-sensitive languages are recursive.

Proof.

Let G be a context-sensitive grammar.

We argue that there exists a Turing machine M accepting $L(G)$.

Context-Sensitive Languages are Recursive

Theorem

Context-sensitive languages are recursive.

Proof.

Let G be a context-sensitive grammar.

We argue that there exists a Turing machine M accepting $L(G)$.

Let $w \in T^*$ be the input word.

Context-Sensitive Languages are Recursive

Theorem

Context-sensitive languages are recursive.

Proof.

Let G be a context-sensitive grammar.

We argue that there exists a Turing machine M accepting $L(G)$.

Let $w \in T^*$ be the input word.

The are finitely words over $V \cup T$ of length $\leq |w|$:

Context-Sensitive Languages are Recursive

Theorem

Context-sensitive languages are recursive.

Proof.

Let G be a context-sensitive grammar.

We argue that there exists a Turing machine M accepting $L(G)$.

Let $w \in T^*$ be the input word.

The are finitely words over $V \cup T$ of length $\leq |w|$:

- M can compute the set $\{\, u \mid S \Rightarrow^* u,\ |u| \leq |w| \,\}$

Context-Sensitive Languages are Recursive

Theorem

Context-sensitive languages are recursive.

Proof.

Let G be a context-sensitive grammar.

We argue that there exists a Turing machine M accepting $L(G)$.

Let $w \in T^*$ be the input word.

The are finitely words over $V \cup T$ of length $\leq |w|$:

- M can compute the set $\{\, u \mid S \Rightarrow^* u,\ |u| \leq |w| \,\}$
- M accepts w if w is among these words.
  (Otherwise M halts in a non-accepting state.)

Context-Sensitive Languages are Recursive

Theorem

Context-sensitive languages are recursive.

Proof.

Let G be a context-sensitive grammar.

We argue that there exists a Turing machine M accepting $L(G)$.

Let $w \in T^*$ be the input word.

The are finitely words over $V \cup T$ of length $\leq |w|$:

- M can compute the set $\{ u \mid S \Rightarrow^* u,\ |u| \leq |w| \}$
- M accepts w if w is among these words.
  (Otherwise M halts in a non-accepting state.)

Then M accepts $L(G)$ and always reaches a halting state. □

Context-Sensitive versus Recursive Languages

Theorem

Not every recursive language is context-sensitive.

Context-Sensitive versus Recursive Languages

Theorem

Not every recursive language is context-sensitive.

Proof.

$\Sigma = \{0, 1\}$. There exists an **injective, computable function**

$$h : \{\, G \mid G \text{ context-sensitive} \,\} \to \{\, 0, 1 \,\}^*$$

such that the **image** of h is **recursive.** For example:

$$h(0) = 010 \qquad h(\to) = 01110 \qquad h(A_i) = 01^{i+4}0$$
$$h(1) = 0110 \qquad h(;) = 011110$$

Context-Sensitive versus Recursive Languages

Theorem

Not every recursive language is context-sensitive.

Proof.

$\Sigma = \{0, 1\}$. There exists an **injective, computable function**

$$h : \{ G \mid G \text{ context-sensitive} \} \to \{ 0, 1 \}^*$$

such that the **image** of h is **recursive.** For example:

$$h(0) = 010 \qquad h(\to) = 01110 \qquad h(A_i) = 01^{i+4}0$$
$$h(1) = 0110 \qquad h(;) = 011110$$

Define $L = \{ h(G) \mid G \text{ context-sensitive } \wedge\ h(G) \notin L(G) \}$.

Context-Sensitive versus Recursive Languages

Theorem

Not every recursive language is context-sensitive.

Proof.

$\Sigma = \{0, 1\}$. There exists an **injective, computable function**

$$h : \{\, G \mid G \text{ context-sensitive} \,\} \to \{\, 0, 1 \,\}^*$$

such that the **image** of h is **recursive.** For example:

$$h(0) = 010 \qquad h(\to) = 01110 \qquad h(A_i) = 01^{i+4}0$$
$$h(1) = 0110 \qquad h(;) = 011110$$

Define $L = \{\, h(G) \mid G \text{ context-sensitive} \;\wedge\; h(G) \notin L(G) \,\}$.
Then L is recursive (by the above assumptions on h).

Context-Sensitive versus Recursive Languages

Theorem

Not every recursive language is context-sensitive.

Proof.

$\Sigma = \{0, 1\}$. There exists an **injective, computable function**

$$h : \{\, G \mid G \text{ context-sensitive} \,\} \to \{\, 0, 1 \,\}^*$$

such that the **image** of h is **recursive.** For example:

$$\begin{array}{lll} h(0) = 010 & h(\to) = 01110 & h(A_i) = 01^{i+4}0 \\ h(1) = 0110 & h(;) = 011110 & \end{array}$$

Define $L = \{\, h(G) \mid G \text{ context-sensitive} \;\wedge\; h(G) \notin L(G) \,\}$.
Then L is recursive (by the above assumptions on h).

Assume $L = L(G_0)$ for a context-sensitive grammar G_0.

Context-Sensitive versus Recursive Languages

Theorem

Not every recursive language is context-sensitive.

Proof.

$\Sigma = \{0, 1\}$. There exists an **injective, computable function**

$$h : \{\, G \mid G \text{ context-sensitive} \,\} \rightarrow \{\, 0, 1 \,\}^*$$

such that the **image** of h is **recursive**. For example:

$$h(0) = 010 \qquad h(\rightarrow) = 01110 \qquad h(A_i) = 01^{i+4}0$$
$$h(1) = 0110 \qquad h(;) = 011110$$

Define $L = \{\, h(G) \mid G \text{ context-sensitive} \wedge h(G) \notin L(G) \,\}$.
Then L is recursive (by the above assumptions on h).

Assume $L = L(G_0)$ for a context-sensitive grammar G_0. Then

$$h(G_0) \in L \iff h(G_0) \notin L(G_0) \iff h(G_0) \notin L$$

Context-Sensitive versus Recursive Languages

Theorem

Not every recursive language is context-sensitive.

Proof.

$\Sigma = \{0, 1\}$. There exists an **injective, computable function**

$$h : \{ G \mid G \text{ context-sensitive} \} \to \{ 0, 1 \}^*$$

such that the **image** of h is **recursive**. For example:

$$h(0) = 010 \qquad h(\to) = 01110 \qquad h(A_i) = 01^{i+4}0$$
$$h(1) = 0110 \qquad h(;) = 011110$$

Define $L = \{ h(G) \mid G \text{ context-sensitive } \wedge\ h(G) \notin L(G) \}$.
Then L is recursive (by the above assumptions on h).

Assume $L = L(G_0)$ for a context-sensitive grammar G_0. Then

$$h(G_0) \in L \iff h(G_0) \notin L(G_0) \iff h(G_0) \notin L$$

Contradiction! $\square$