Universiteit Utrecht

[Faculty of **Science**
Information and Computing Sciences]

Talen en Compilers

2022 - 2023, period 2

David van Balen

Department of Information and Computing Sciences
Utrecht University

2023-01-13

12. Pumping lemma's and context-free languages

This lecture

wooclap questions

Pumping lemma's and context-free languages

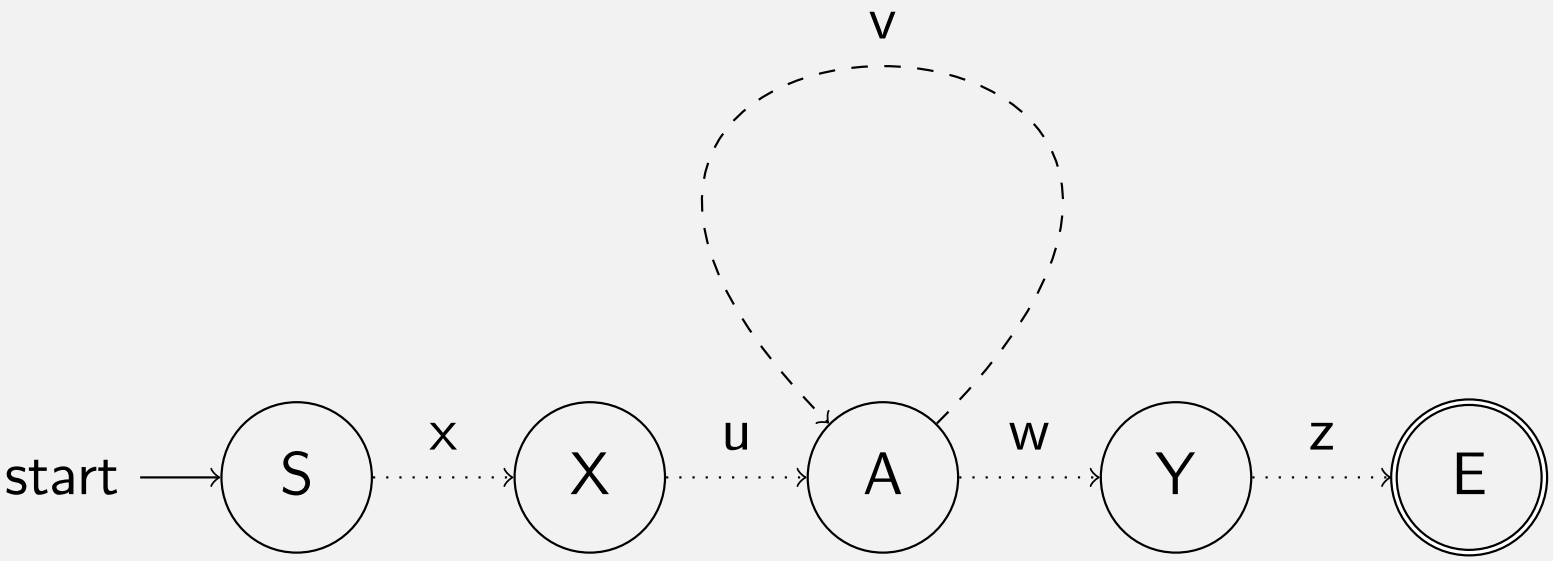
- Pumping lemma for regular languages
- Pumping lemma for context-free languages

12.1 Pumping lemma for regular languages

Recap

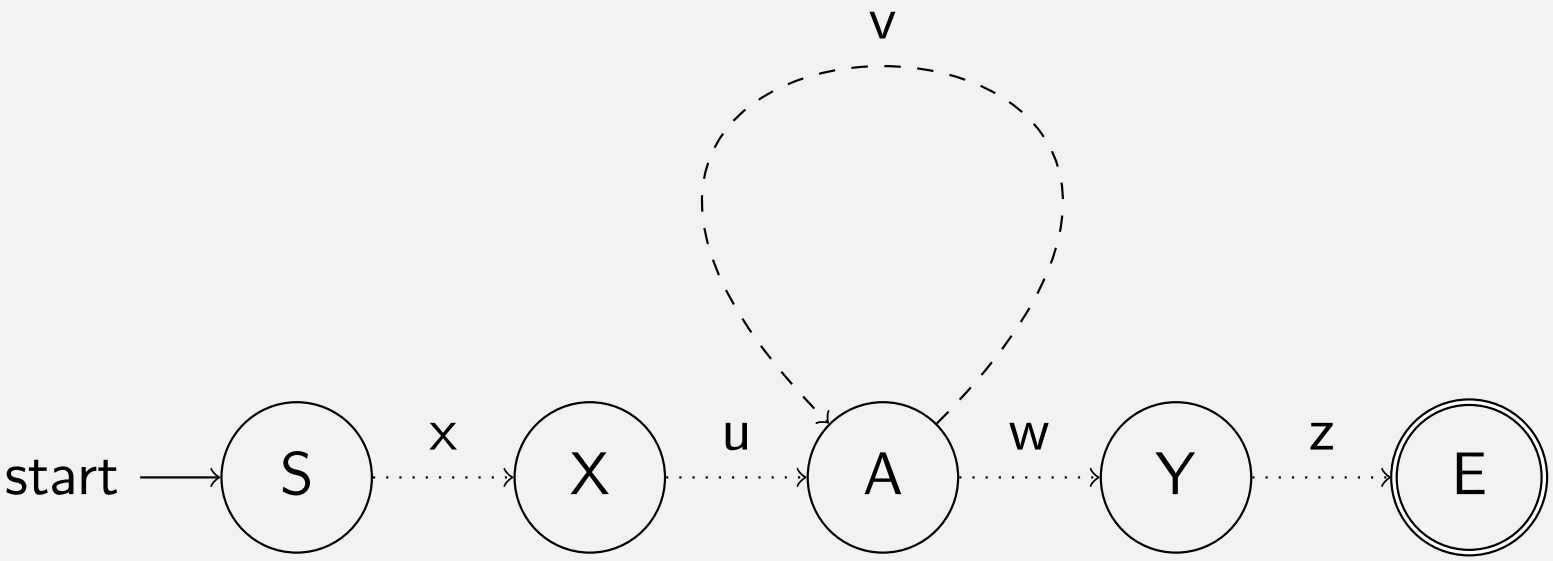
- Assume we have an accepted word xyz where subword y is of at least length n.

Recap



- Assume we have an accepted word xyz where subword y is of at least length n.
- Then y has to be of form uvw where v is not empty and corresponds to a loop.

Recap



- Assume we have an accepted word xyz where subword y is of at least length n.
- Then y has to be of form uvw where v is not empty and corresponds to a loop.
- All words of the form xuv^iwz for $i \in \mathbb{N}$ are accepted.

Recap

Pumping Lemma for regular languages

For every regular language L,

Recap

Pumping Lemma for regular languages

For every regular language L,
there exists an $n \in \mathbb{N}$

- (corresponding to the number of states in the automaton)

Recap

Pumping Lemma for regular languages

For every regular language L,
there exists an $n \in \mathbb{N}$

such that for every word xyz in L with $|y| \geqslant n$,

Recap

Pumping Lemma for regular languages

For every regular language L,
there exists an n $\in \mathbb{N}$

such that for every word xyz in L with $|y| \geqslant n$,
we can split y into three parts, $y = uvw$, with $|v| > 0$,

Recap

Pumping Lemma for regular languages

For every regular language L,
there exists an $n \in \mathbb{N}$

such that for every word xyz in L with $|y| \geqslant n$,
we can split y into three parts, $y = uvw$, with $|v| > 0$,
such that for every $i \in \mathbb{N}$, we have $xuv^iwz \in L$.

Recap

- For **every** natural number n,
  - because you don't know what the value of n is
- find a word xyz in L with $|y| \geqslant n$ (**you choose** the word),
- such that for **every** splitting $y = uvw$ with $|v| > 0$,
  - because you don't know where the loop may be
- there exists a number i (**you choose** the number),
- such that $xuv^iwz \notin L$ (you have to **prove** it).

12.2 Pumping lemma for context-free languages

Context-free grammars

A context-**free** grammar consists of a sequence of productions:

$$N \to x$$

- the **left hand side** is always a **nonterminal**,
- the right hand side is any sequence of terminals and nonterminals.

One nonterminal of the grammar is the start symbol.

Context-sensitive grammars

Context-**sensitive** grammars drop the restriction on the left hand side:

$$a\ N\ b \rightarrow x$$

Context-sensitive grammars

Context-**sensitive** grammars drop the restriction on the left hand side:

> $a\ N\ b \rightarrow x$

Context-sensitive grammars are as **powerful** as any other computing formalism:

- Turing machines,
- λ-calculus.

Not interesting from a parsing perspective.

The strategy – revisited

If we want to prove that a certain language is **not context-free**, we can apply the same strategy as for regular languages:

- we expose a limitation in the formalism (in this case, in the concept of context-free grammars);
- from this limitation, we derive a property that all languages in the class (in this case, context-free languages) must have;
- therefore, if a language does not have that property, it cannot be in the class.

The strategy – revisited

If we want to prove that a certain language is **not context-free**, we can apply the same strategy as for regular languages:

- we expose a limitation in the formalism (in this case, in the concept of context-free grammars);
- from this limitation, we derive a property that all languages in the class (in this case, context-free languages) must have;
- therefore, if a language does not have that property, it cannot be in the class.

This time, we analyze parse trees rather than finite state automata.

Grammars and parse trees

For every word in the language, there is a parse tree.

We observe:

Grammars and parse trees

For every word in the language, there is a parse tree.

We observe:

- We can produce parse trees of arbitrary depth if we find words in the language that are long enough, because the number of children per node is bounded by the maximum length of a right hand side of a production.

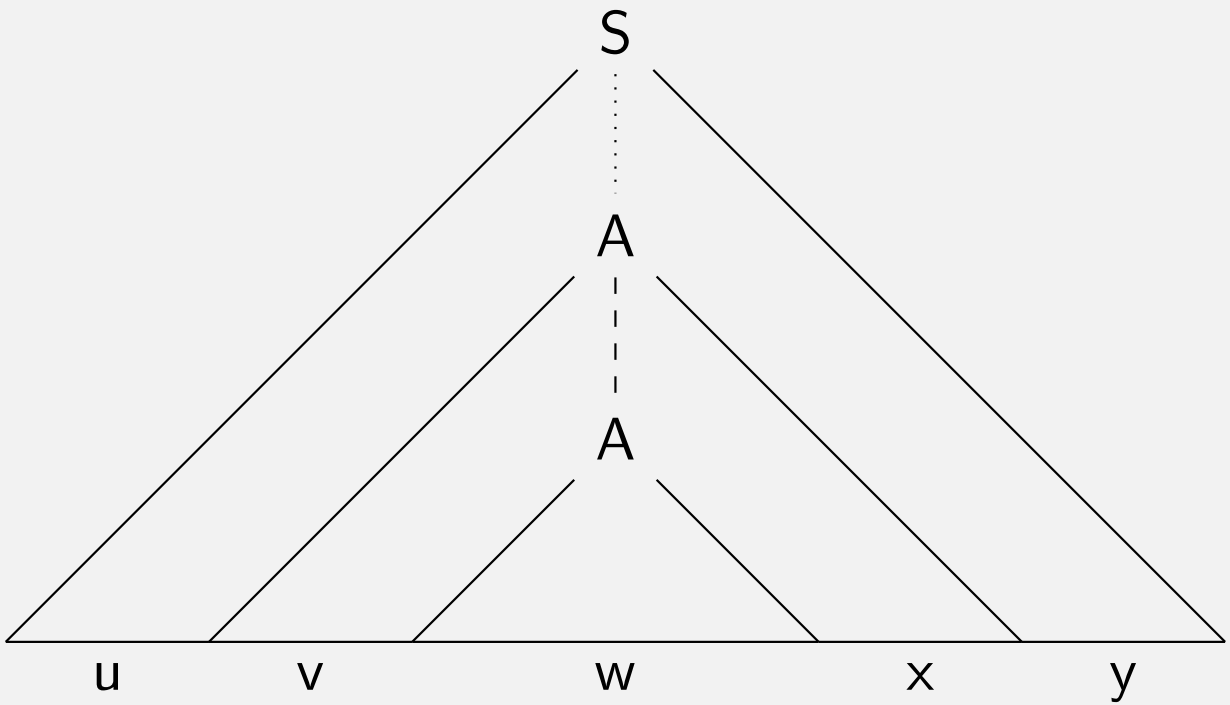
Grammars and parse trees

For every word in the language, there is a parse tree.

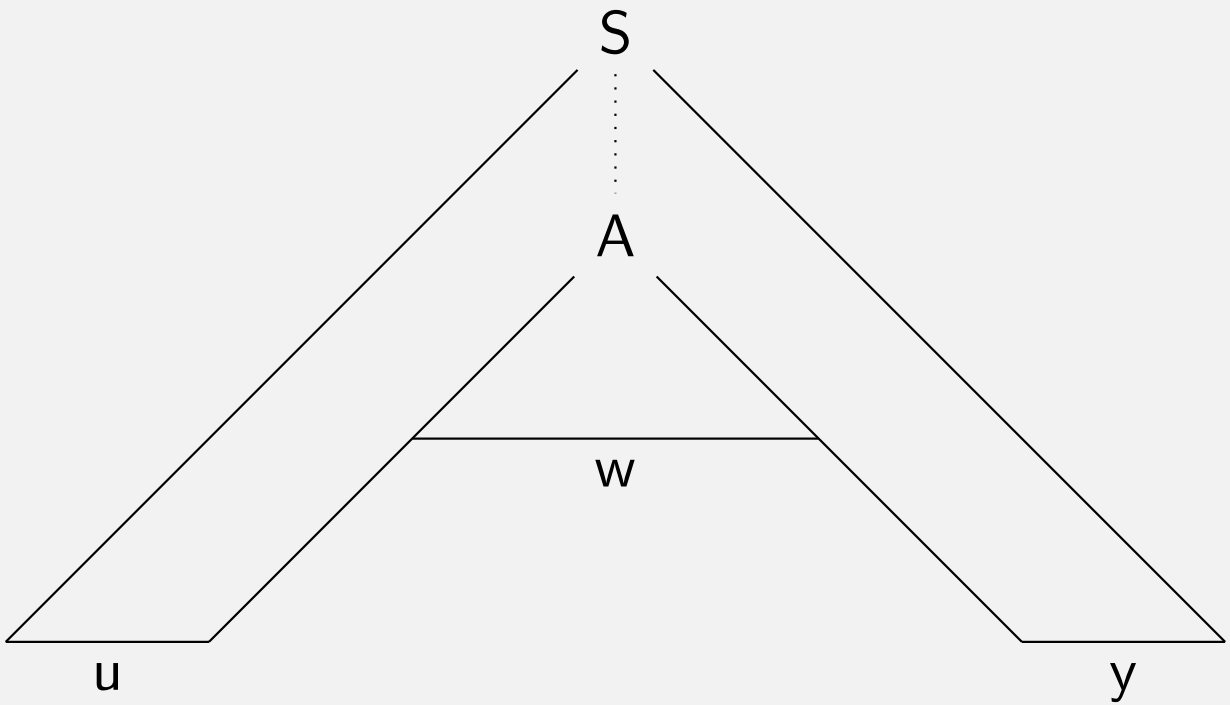
We observe:

- We can produce parse trees of arbitrary depth if we find words in the language that are long enough, because the number of children per node is bounded by the maximum length of a right hand side of a production.
- Once a path from a leaf to the root has more than n internal nodes, where n is the number of nonterminals in the grammar, one nonterminal has to occur twice on such a path.

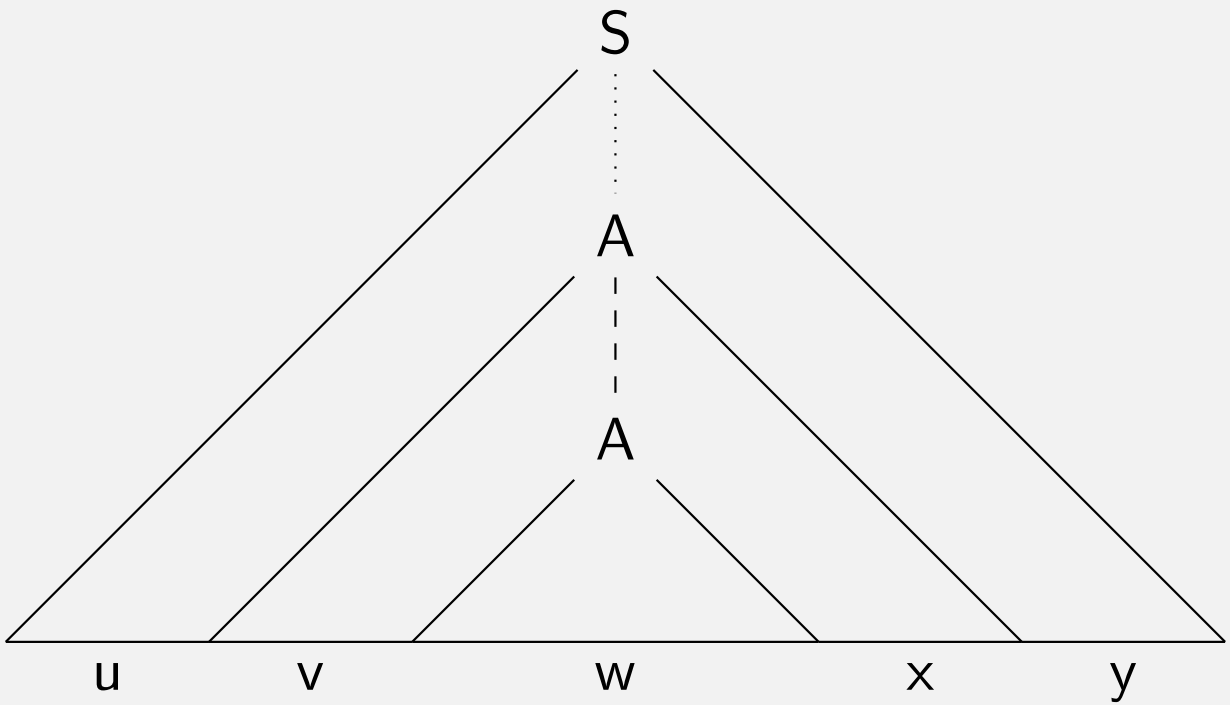
The situation



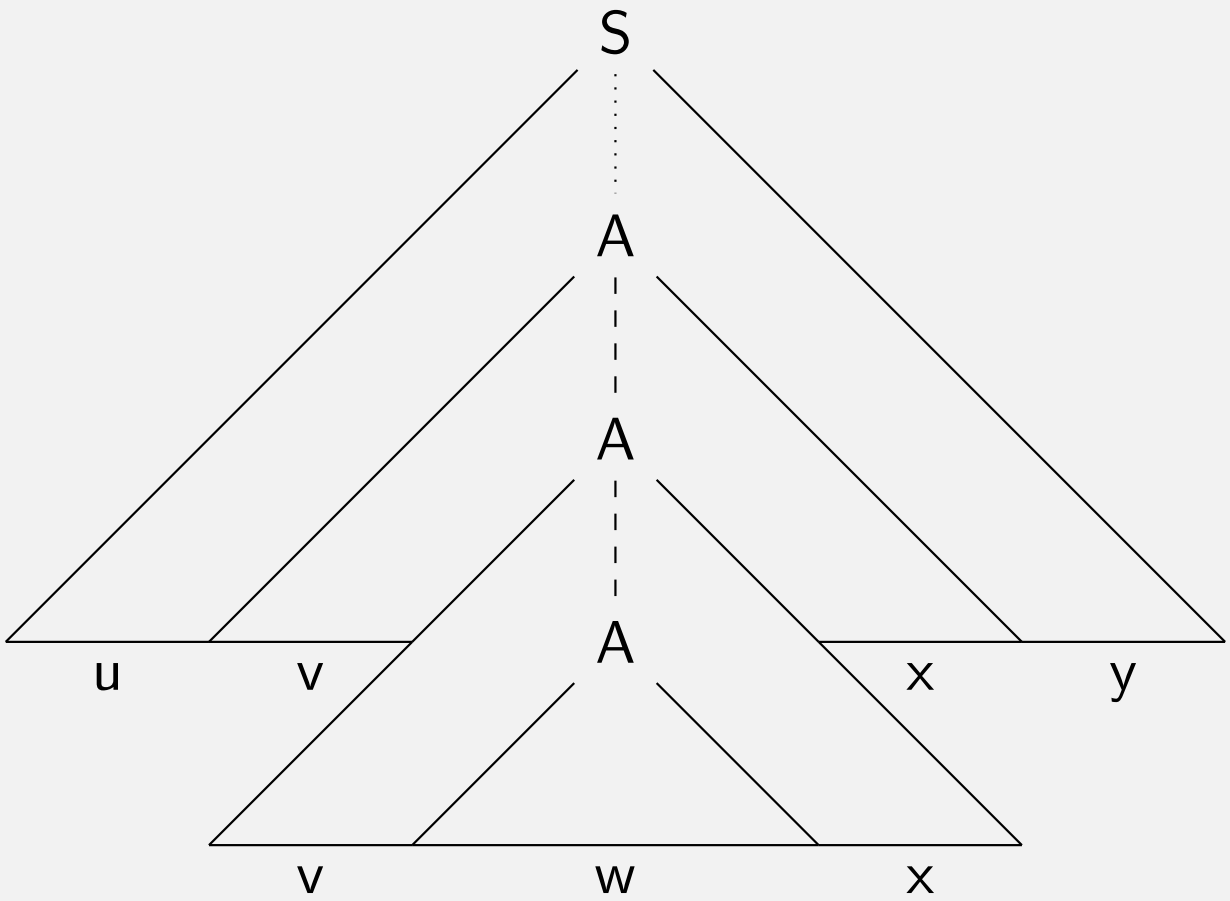
The situation



The situation



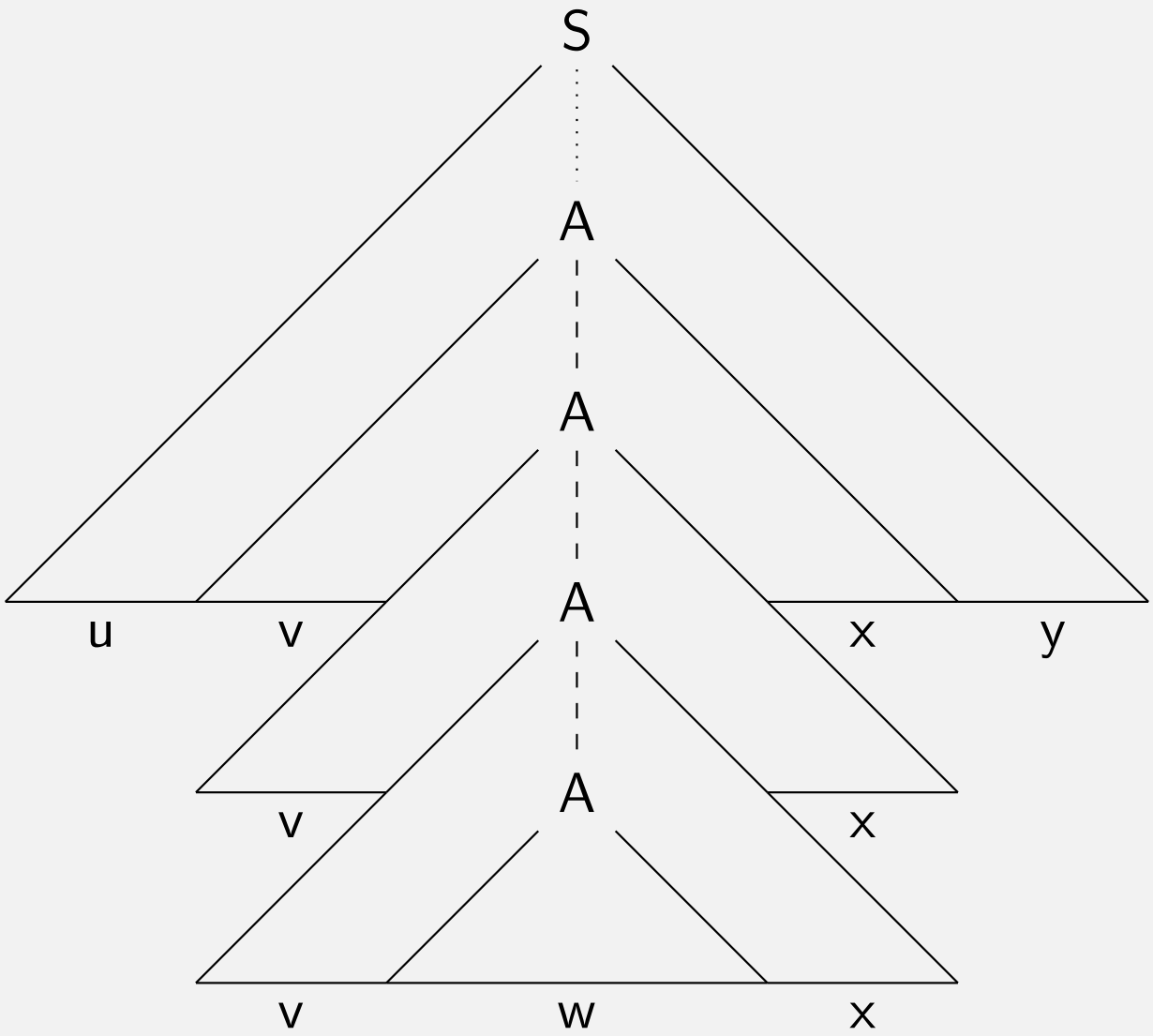
The situation



Universiteit Utrecht

[Faculty of **Science**
Information and Computing Sciences]

The situation



Universiteit Utrecht

[Faculty of **Science**
Information and Computing Sciences]

The situation – contd.

If the word is long enough, we have a derivation of the form

$$S \Rightarrow^* uAy \Rightarrow^* uvAxy \Rightarrow^* uvwxy$$

where $|vx| > 0$.

The situation – contd.

If the word is long enough, we have a derivation of the form

$$S \Rightarrow^* uAy \Rightarrow^* uvAxy \Rightarrow^* uvwxy$$

where $|vx| > 0$.

Because the grammar is context-free, this implies that

$$A \Rightarrow^* vAx$$
$$A \Rightarrow^* w$$

The situation – contd.

If the word is long enough, we have a derivation of the form

$$S \Rightarrow^* uAy \Rightarrow^* uvAxy \Rightarrow^* uvwxy$$

where $|vx| > 0$.

Because the grammar is context-free, this implies that

$$A \Rightarrow^* vAx$$
$$A \Rightarrow^* w$$

We can thus derive

$$S \Rightarrow^* uAy \Rightarrow^* uv^iwx^iy$$

for any $i \in \mathbb{N}$.

The lemma

Pumping lemma for context-free languages

For every context-free language L,

The lemma

Pumping lemma for context-free languages

For every context-free language L,

- there exists a number n $\in \mathbb{N}$ such that

The lemma

Pumping lemma for context-free languages

For every context-free language L,

- there exists a number $n \in \mathbb{N}$ such that
- for every word $z \in L$ with $|z| \geqslant n$,

The lemma

Pumping lemma for context-free languages

For every context-free language L,

- there exists a number $n \in \mathbb{N}$ such that
- for every word $z \in L$ with $|z| \geqslant n$,
- we can split z into five parts, $z = uvwxy$, with $|vx| > 0$ and $|vwx| \leqslant n$, such that

The lemma

Pumping lemma for context-free languages

For every context-free language L,

- there exists a number $n \in \mathbb{N}$ such that
- for every word $z \in L$ with $|z| \geqslant n$,
- we can split z into five parts, $z = uvwxy$, with $|vx| > 0$ and $|vwx| \leqslant n$, such that
- for every $i \in \mathbb{N}$, we have $uv^iwx^iy \in L$.

The lemma

Pumping lemma for context-free languages

For every context-free language L,

- there exists a number $n \in \mathbb{N}$ such that
- for every word $z \in L$ with $|z| \geqslant n$,
- we can split z into five parts, $z = uvwxy$, with $|vx| > 0$ and $|vwx| \leqslant n$, such that
- for every $i \in \mathbb{N}$, we have $uv^iwx^iy \in L$.

The n lets us limit the size of the part that gets pumped, similar to how the pumping lemma for regular languages lets us choose the subword that contains the loop.

Using the pumping lemma

- For **every** of number n,
- find a word z in L with $|z| \geqslant n$ (**you choose** the word),
- such that for **every** splitting $z = uvwxy$ with $|vx| > 0$ and $|vwx| \leqslant n$,
- there exists a number i (**you choose** the number),
- such that $uv^iwx^iy \notin L$ (you have to **prove** it).

An example

Theorem

The language $L = \{a^m b^m c^m \mid m \in \mathbb{N}\}$ is not context-free.

An example

Theorem

The language $L = \{a^m b^m c^m \mid m \in \mathbb{N}\}$ is not context-free.

Let n be any number.

An example

Theorem

The language $L = \{a^m b^m c^m \mid m \in \mathbb{N}\}$ is not context-free.

Let n be any number.

We then consider the word $z = a^n b^n c^n$.

An example

Theorem

The language $L = \{a^m b^m c^m \mid m \in \mathbb{N}\}$ is not context-free.

Let n be any number.

We then consider the word $z = a^n b^n c^n$.

From the pumping lemma, we learn that we can pump z, and that the part that gets pumped is smaller than n.

An example

Theorem

The language $L = \{a^m b^m c^m \mid m \in \mathbb{N}\}$ is not context-free.

Let n be any number.

We then consider the word $z = a^n b^n c^n$.

From the pumping lemma, we learn that we can pump z, and that the part that gets pumped is smaller than n.

The part being pumped can thus not contain a's, b's **and** c's at the same time, and is not empty either. In all these cases, we pump out of the language (for any $i \neq 1$).

Normal forms

Context-free grammars can be wildly complex, in general.

But all of them can be brought into more normalised forms.

- We call them **normal forms**.

We get to them by applying **grammar transformations** (see lecture 4).

Chomsky Normal Form

A context-free grammar is in **Chomsky Normal Form** if each production rule has one of these forms:

$$A \to B\ C$$
$$A \to \mathtt{x}$$
$$S \to \varepsilon$$

where A, B, and C are nonterminals, $\mathtt{x}$ is a terminal, and S is the start symbol of the grammar. Also, B and C cannot be S.

Chomsky Normal Form

A context-free grammar is in **Chomsky Normal Form** if each production rule has one of these forms:

$$A \to B\ C$$
$$A \to \mathtt{x}$$
$$S \to \varepsilon$$

where A, B, and C are nonterminals, $\mathtt{x}$ is a terminal, and S is the start symbol of the grammar. Also, B and C cannot be S.

- No rule produces ε except (possibly) from the start.
- No chain rules of the form $A \to B$.
- Parse trees are always binary.

Greibach Normal Form

A context-free grammar is in **Greibach Normal Form** if each production rule has one of these forms:

$$\mathsf{A} \to \mathtt{x}\mathsf{A}_1\mathsf{A}_2 \ldots A_n$$
$$\mathsf{S} \to \varepsilon$$

where A, A_1, ..., A_n are nonterminals ($n \geqslant 0$), $\mathtt{x}$ is a terminal, and S is the start symbol of the grammar and does not occur in any right hand side.

Greibach Normal Form

A context-free grammar is in **Greibach Normal Form** if each production rule has one of these forms:

$$\mathrm{A} \to \mathtt{x}\mathrm{A}_1\mathrm{A}_2 \ldots A_n$$
$$\mathrm{S} \to \varepsilon$$

where A, A_1, ..., A_n are nonterminals ($n \geqslant 0$), $\mathtt{x}$ is a terminal, and S is the start symbol of the grammar and does not occur in any right hand side.

- At most one rule produces ε, and only from the start.
- No left recursion.
- A derivation of a word of length n has exactly n rule applications (except ε).
- Generalizes GNF for regular grammars (where $n \leqslant 1$)