Universiteit Utrecht

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

Talen en Compilers

2018 - 2019, period 2

Alejandro Serrano Mena

Department of Information and Computing Sciences
Utrecht University

12. Context-free languages

This lecture

Context-free languages

- Context-free grammars
- Pushdown automata (PDA)
- PDA vs. context-free grammars
- Closure properties
- Pumping lemma for context-free languages

12.1 Context-free grammars

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:

> $a\ N\ b \rightarrow x$

The context around N "controls" whether the production may apply or not.

Context-sensitive grammars

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

> a N b $\rightarrow$ x

The context around N "controls" whether the production may apply or not.

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

- Turing machines,
- λ-calculus.

Not interesting from a parsing perspective.

Normal forms

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

But all of them can be brought into more simplified 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.

Chomsky Normal Form

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

$$
\begin{aligned}
&A \to B\ C \\
&A \to \mathtt{x} \\
&S \to \varepsilon
\end{aligned}
$$

where A, B, and C are nonterminals, $\mathtt{x}$ is a terminal, and S is the start symbol of the grammar.

- No rule produces ε except 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:

$$
\begin{aligned}
&A \to \mathtt{x}BC\ldots Z\\
&A \to \mathtt{x}\\
&S \to \varepsilon
\end{aligned}
$$

where A, B, ..., Z are nonterminals, $\mathtt{x}$ is a terminal, and S is the start symbol of the grammar.

Greibach Normal Form

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

$$A \to \mathtt{x}BC\ldots Z$$
$$A \to \mathtt{x}$$
$$S \to \varepsilon$$

where A, B, ..., Z are nonterminals, $\mathtt{x}$ is a terminal, and S is the start symbol of the grammar.

- No rule produces ε except from the start.
- No left recursion.
- A derivation of a word of length n has exactly n rule applications.

Greibach Normal Form for regular grammars

Regular grammars have productions of the form, where x is a sequence of terminals:

$$A \to xB$$
$$A \to x$$

Grammars in GNF have productions of the form, where $\mathtt{x}$ is a single terminal:

$$A \to \mathtt{x}BC \ldots Z$$
$$A \to \mathtt{x}$$
$$S \to \varepsilon$$

Putting these together we obtain the definition of GNF for regular languages from the previous lecture.

12.2 Pushdown automata (PDA)

The question

Is there something similar to finite state automata, but for context-free languages?

The question

Is there something similar to finite state automata, but for context-free languages?

Yes, **pushdown automata** (PDA)!

The question

Is there something similar to finite state automata, but for context-free languages?

Yes, **pushdown automata** (PDA)!

Concisely, pushdown automata = finite state automata + stack.

Pushdown automata (PDA)

An extension of non-deterministic finite automata.

- We have an additional alphabet Γ of **stack symbols**.
- Transitions take the general form:

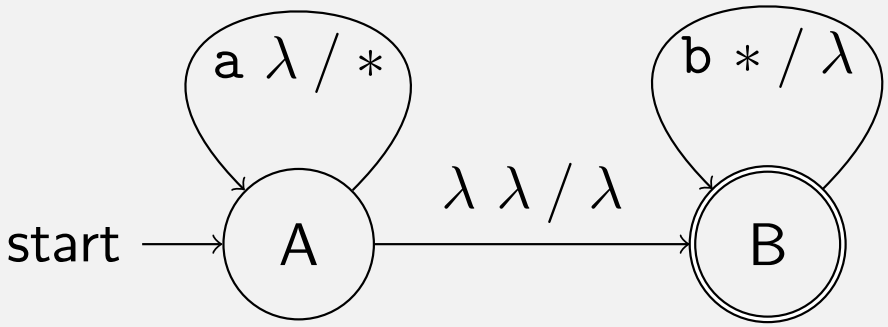
  $$\mathsf{d} :: \mathsf{Q} \to \mathsf{Maybe}\ \mathsf{X} \to \mathsf{Maybe}\ \Gamma \to \mathsf{Set}\ (\mathsf{Q}, \mathsf{Maybe}\ \Gamma)$$

  where we represent absence using the symbol λ.

  - a A / B: if A is the top of the stack, pop it and push B,
  - a λ / B: push a B regadless of the state of the stack,
  - a A / λ: transition only if the top of the stack has an A, pop it, but do not push anything.

A word is accepted only if the **stack** finishes **empty**.

PDA for $\{a^n b^n \mid n \in \mathbb{N}\}$



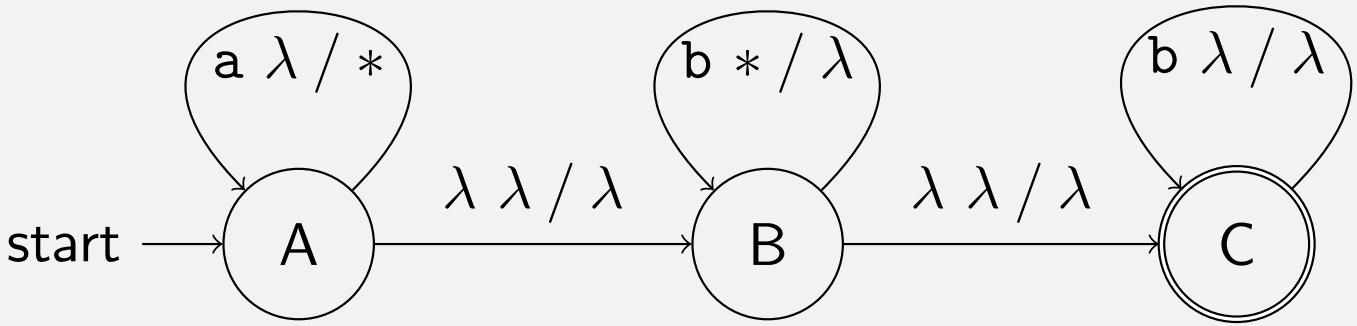
- The stack alphabet is $\{*\}$.
- We push $*$ for each a we find, and then we pop for each b.
- Remember that the word in only accepted if at the end the stack is empty.

PDA for $\{a^m b^n \mid m, n \in \mathbb{N}, m < n\}$

Try to write the PDA by modifying the previous one.

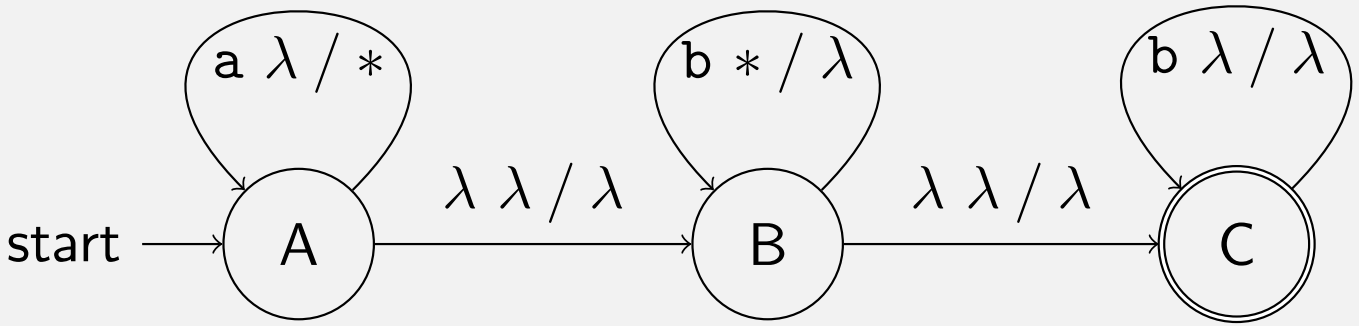
PDA for $\{a^m b^n \mid m, n \in \mathbb{N}, m < n\}$

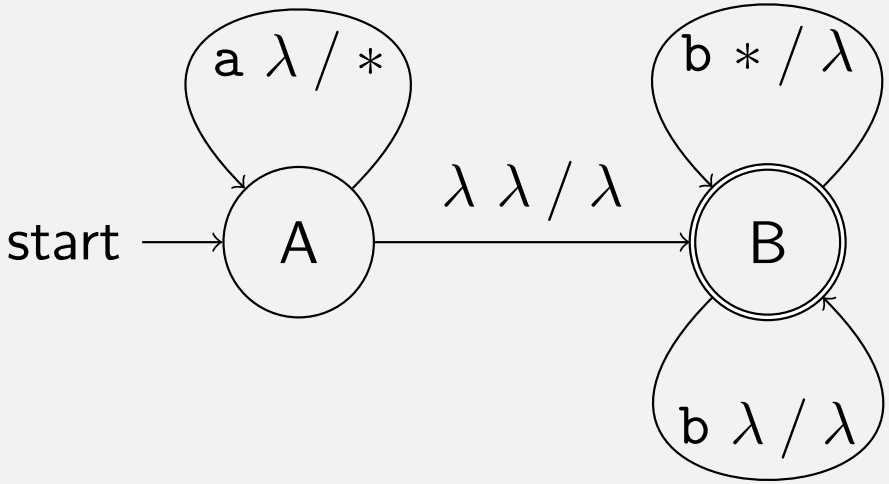
Try to write the PDA by modifying the previous one.



PDA for $\{a^m b^n \mid m, n \in \mathbb{N}, m < n\}$

Try to write the PDA by modifying the previous one.





PDA with extended transitions

Transitions may push more then one symbol to the stack:

$$\mathsf{d} :: \mathsf{Q} \to \mathsf{Maybe}\ \mathsf{X} \to \mathsf{Maybe}\ \Gamma \to \mathsf{Set}\ (\mathsf{Q}, [\Gamma])$$

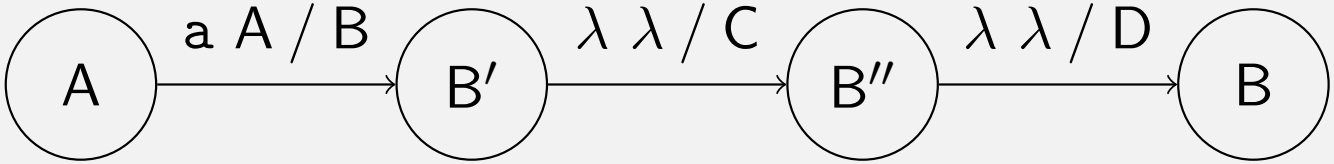
PDA with extended transitions

Transitions may push more then one symbol to the stack:

$$\mathsf{d} :: \mathsf{Q} \to \mathsf{Maybe}\ \mathsf{X} \to \mathsf{Maybe}\ \Gamma \to \mathsf{Set}\ (\mathsf{Q}, [\Gamma])$$

We can transform such an extended PDA into a regular one:

- For each transition which pushes more than one symbol, introduce new states which consecutively push one symbol.



12.3 PDA vs. context-free grammars

From context-free grammars to PDA

For each context-free grammar, we can build a PDA which accepts the same language.

From context-free grammars to PDA

For each context-free grammar, we can build a PDA which accepts the same language.

Construction

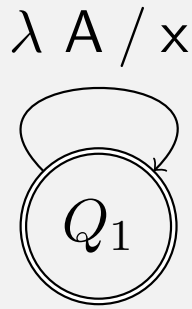
- The input alphabet consists of the terminals.
- The stack alphabet consists of the terminals and nonterminals.
- We have two states, a starting Q_0 and an accepting Q_1.

From context-free grammars to PDA

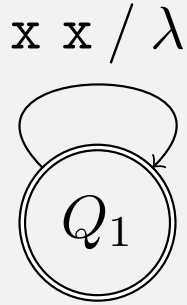
Construction – transitions

- Start the computation:

- For each rule $A \to x$:



- For each terminal $\mathtt{x}$:



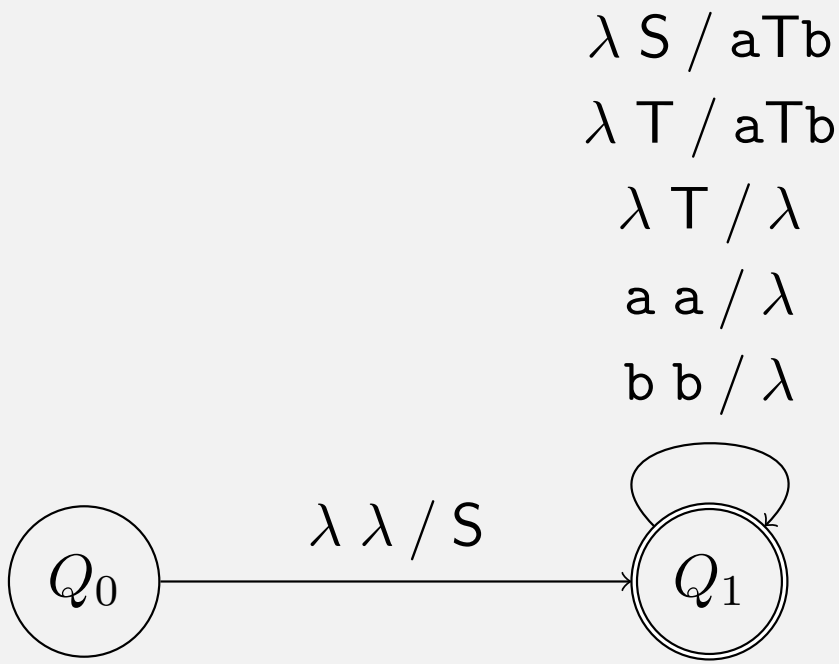
Expand-Match parsing algorithm

This algorithm is called **Expand-Match** in the Lecture Notes.

1. First we always push the start symbol S.
2. In each step, we do one of the following:
   - **Expand** If the top symbol on the stack is a nonterminal, we replace it with a corresponding right hand side of a production.
   - **Match** If the top symbol on the stack is a terminal and the first symbol of the input matches, we remove the symbol from both stack and input.
3. If the stack is empty after all the input, we succeed.

PDA for $\{a^n b^n \mid n \in \mathbb{N}, n > 0\}$, using Expand-Match

$$S \to aTb$$
$$T \to aTb \mid \varepsilon$$



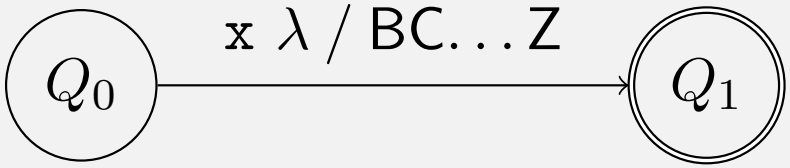
From GNF to PDA

If the grammar is in GNF, we can do a bit better.

Modified construction – transitions

- For each rule with the start symbol on the left

  $S \to \mathtt{x}BC\ldots Z$



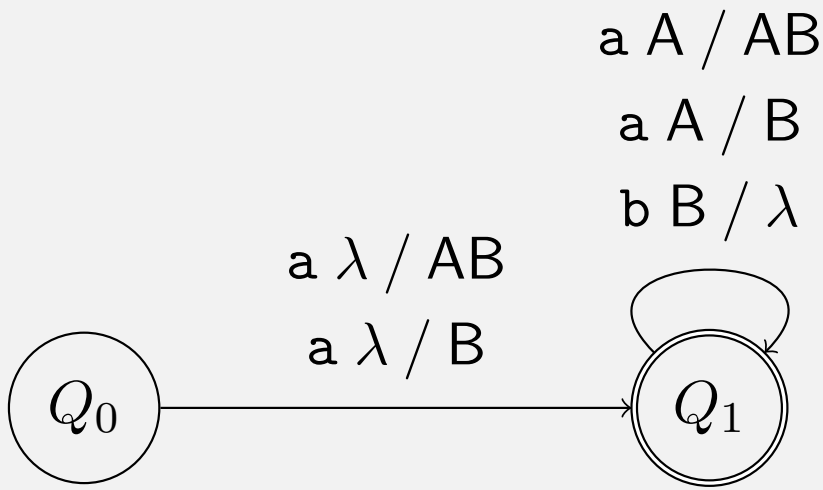
- For each rule with another symbol on the left

  $A \to \mathtt{x}BC\ldots Z$

PDA for $\{a^n b^n \mid n \in \mathbb{N}, n > 0\}$, from a grammar

Greibach N.F.:

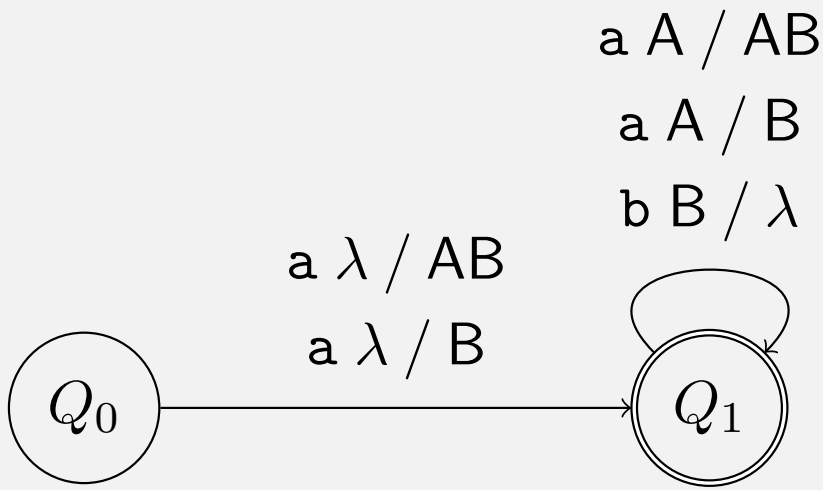
$$
\begin{aligned}
S &\to aAB \mid aB \\
A &\to aAB \mid aB \\
B &\to b
\end{aligned}
$$



PDA for $\{a^n b^n \mid n \in \mathbb{N}, n > 0\}$, from a grammar

Greibach N.F.:

$$
\begin{aligned}
S &\to aAB \mid aB \\
A &\to aAB \mid aB \\
B &\to b
\end{aligned}
$$

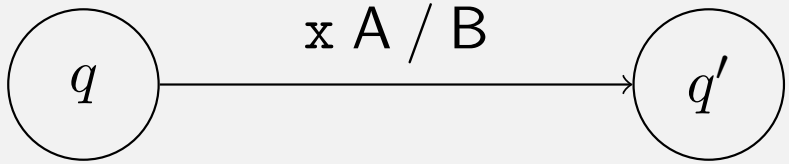


At every step we **consume** a character

- Less non-determinism $\implies$ fewer options to search

From PDA to context-free grammars

Idea of the construction

- The nonterminals of the grammars are triples $\langle\, q, A, q' \,\rangle$, where q and q' are states of the PDA and A a member of the stack alphabet or λ.
  - Represents a transition which begins in q, ends in q', and removes A from the stack.
- For each transition 

we add the following rule for each $q'' \in Q$

$$\langle\, q, A, q' \,\rangle \to x \langle\, q', B, q'' \,\rangle$$

- For each state $q \in Q$ we add the following rule:

$$\langle\, q, \lambda, q \,\rangle \to \varepsilon$$

12.4 Closure properties

Summary

Context-free languages are closed under:

- union,
- concatenation,
- star-closure.

But they are **not** closed under:

- intersection,
- complement.

Proving closure

The proofs are very similar to those of regular languages:

- **Union**: at the beginning move to one start state non-deterministically.
- **Concatenation**: move from accepting states of the first to initial states of the second.
  - This works because each PDA must leave the stack empty.
- **Star-closure**: link the accepting states to the initial states.

Deterministic PDA

The PDA presented here are **non-deterministic**.

- Deterministic PDA have at most one choice per combination of input symbol and top of the stack.

Question

Are deterministic PDA as powerful as non-deterministic ones?

Deterministic PDA

The PDA presented here are **non-deterministic**.

- Deterministic PDA have at most one choice per combination of input symbol and top of the stack.

Question

Are deterministic PDA as powerful as non-deterministic ones?

No, they accept a **smaller** set of languages.

12.5 Pumping lemma for context-free languages

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 context-free grammars 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.

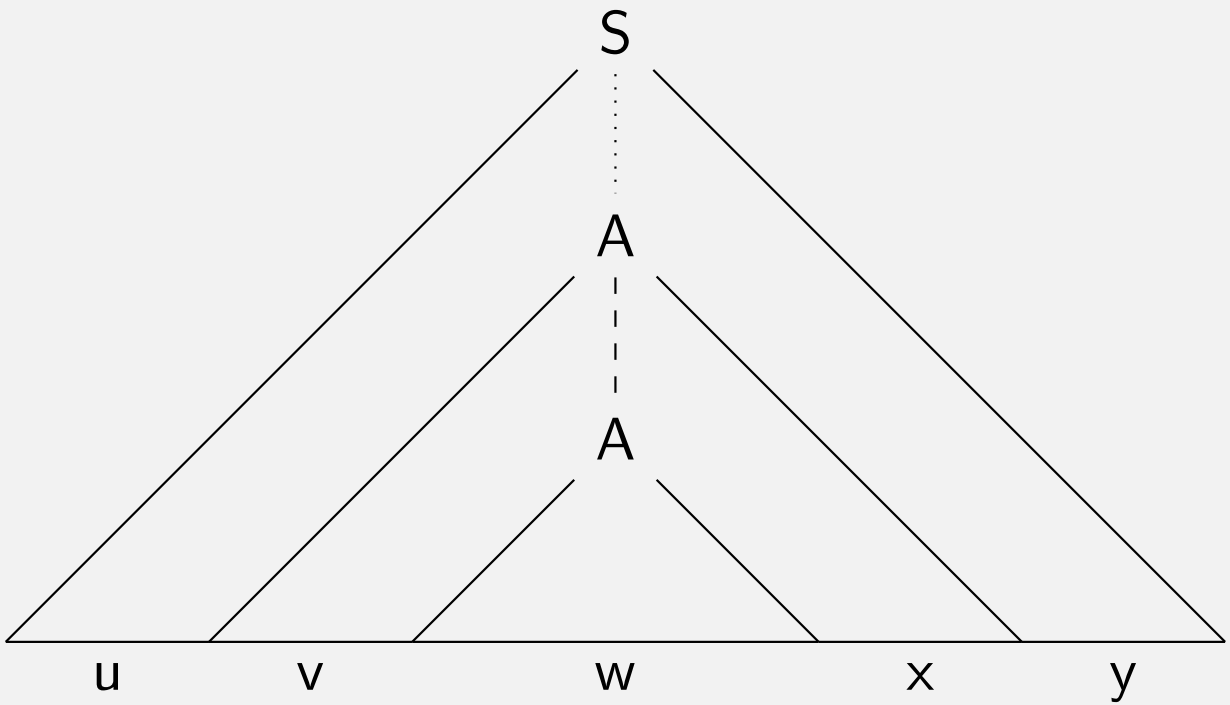
Grammars and parse trees

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

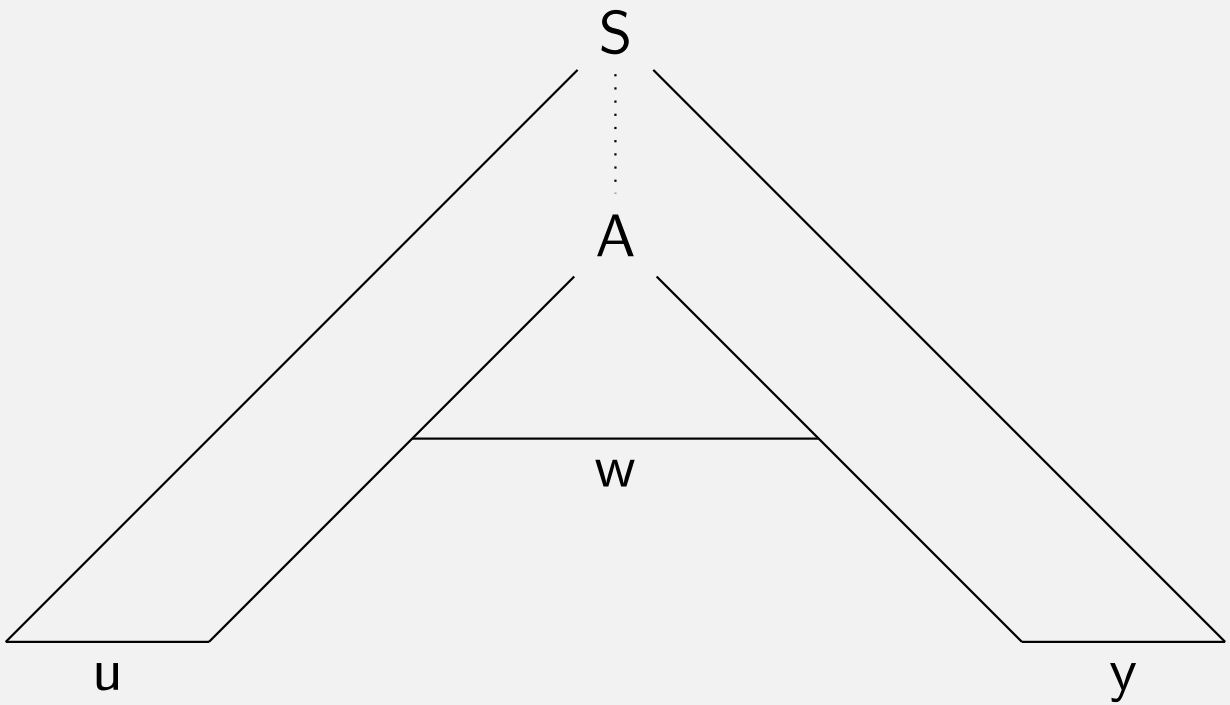
We observe:

- 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.

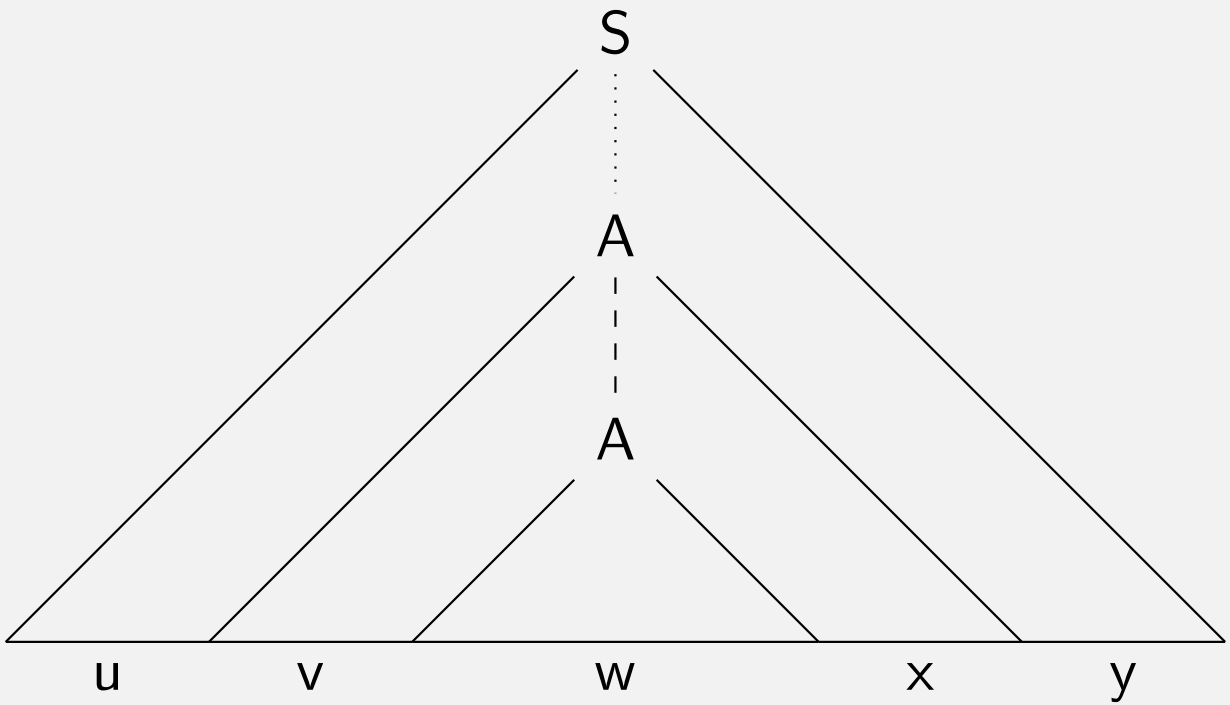
The situation



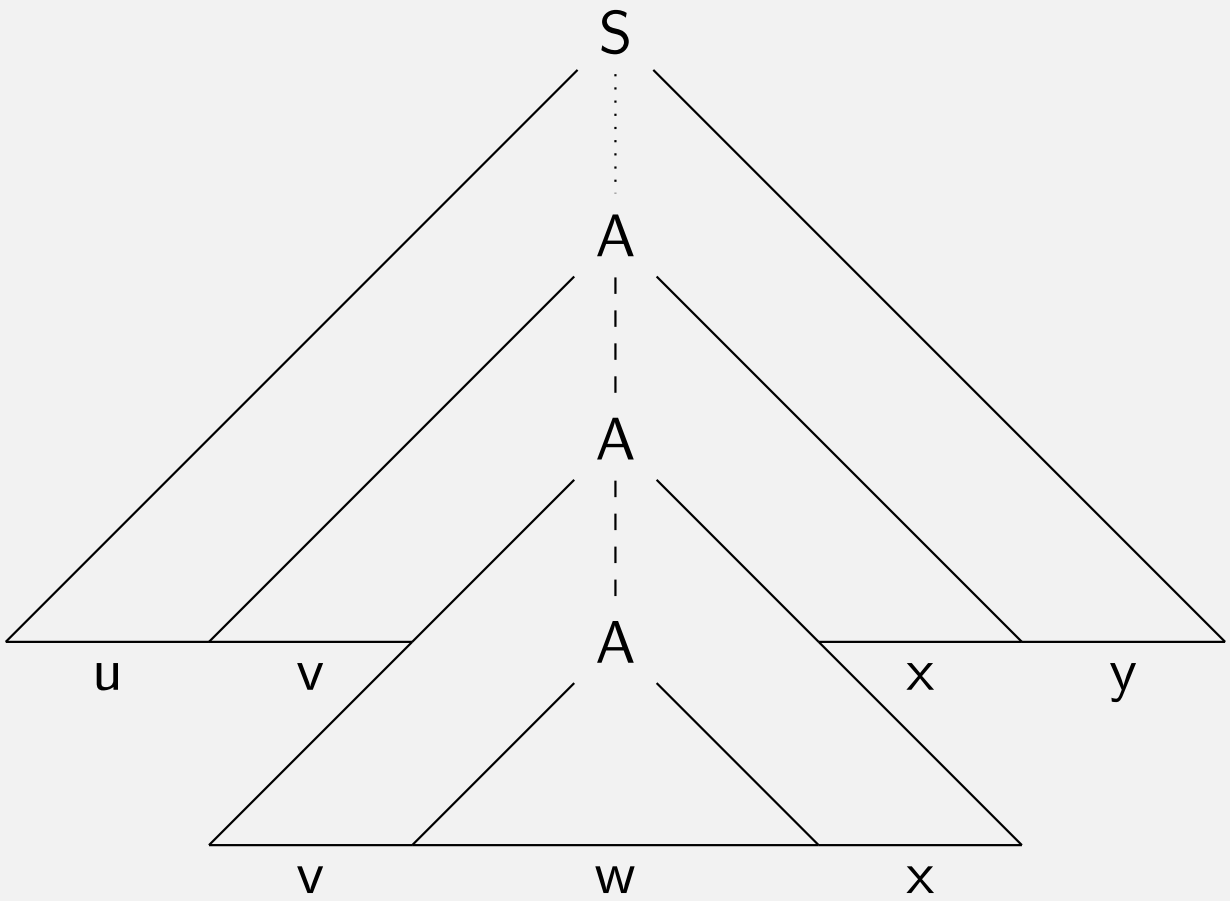
The situation



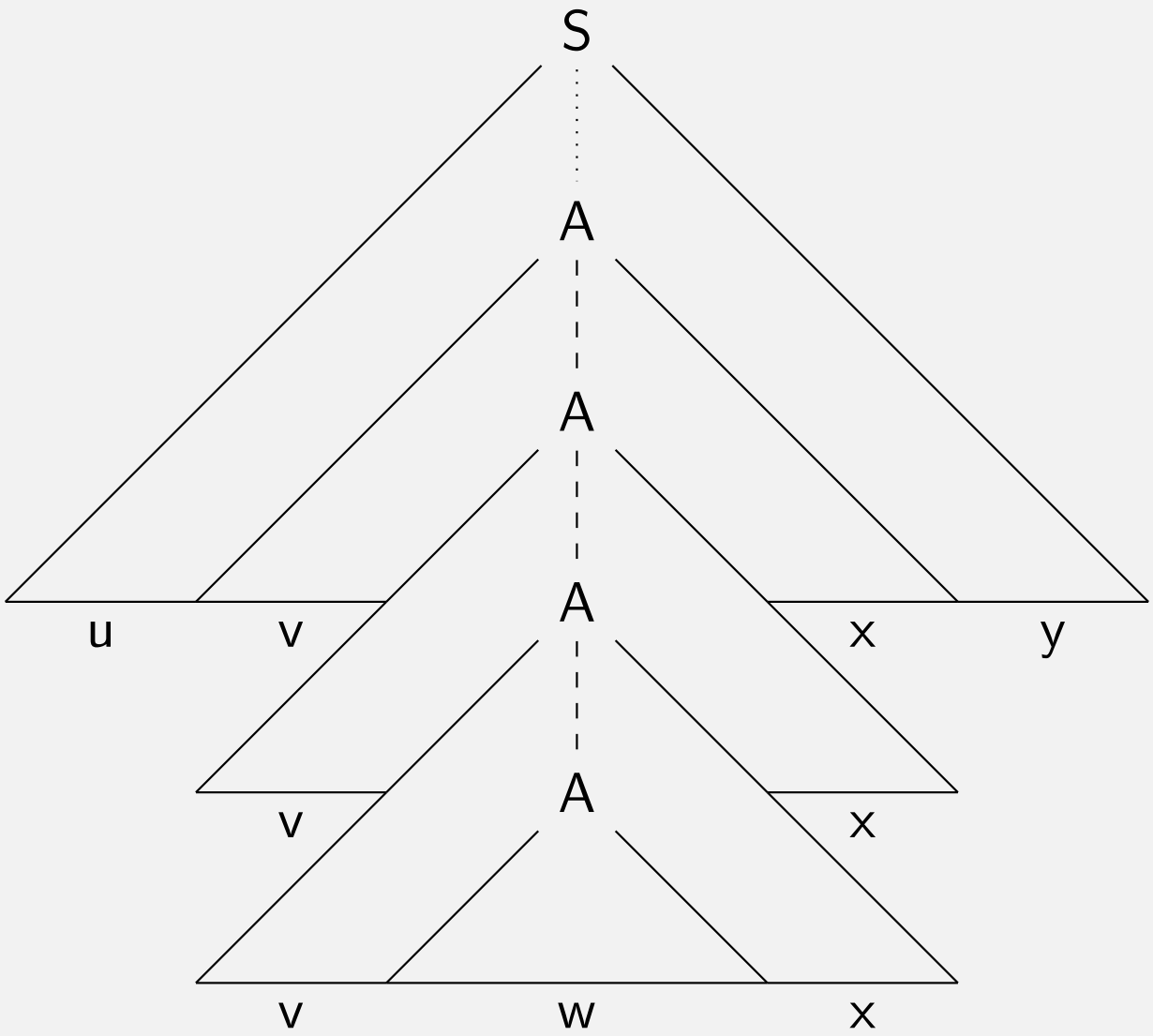
The situation



The situation



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 exist numbers $c, d \in \mathbb{N}$ (determined by the maximum number of nonterminals on the right hand side of a production and the total number of nonterminals) such that

The lemma

Pumping lemma for context-free languages

For every context-free language L,

- there exist numbers $c, d \in \mathbb{N}$ (determined by the maximum number of nonterminals on the right hand side of a production and the total number of nonterminals) such that
- for every word $z \in L$ with $|z| \geqslant c$,

The lemma

Pumping lemma for context-free languages

For every context-free language L,

- there exist numbers $c, d \in \mathbb{N}$ (determined by the maximum number of nonterminals on the right hand side of a production and the total number of nonterminals) such that
- for every word $z \in L$ with $|z| \geqslant c$,
- we can split z into five parts, $z = uvwxy$, with $|vx| > 0$ and $|vwx| \leqslant d$, such that

The lemma

Pumping lemma for context-free languages

For every context-free language L,

- there exist numbers $c, d \in \mathbb{N}$ (determined by the maximum number of nonterminals on the right hand side of a production and the total number of nonterminals) such that
- for every word $z \in L$ with $|z| \geqslant c$,
- we can split z into five parts, $z = uvwxy$, with $|vx| > 0$ and $|vwx| \leqslant d$, 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 exist numbers $c, d \in \mathbb{N}$ (determined by the maximum number of nonterminals on the right hand side of a production and the total number of nonterminals) such that
- for every word $z \in L$ with $|z| \geqslant c$,
- we can split z into five parts, $z = uvwxy$, with $|vx| > 0$ and $|vwx| \leqslant d$, such that
- for every $i \in \mathbb{N}$, we have $uv^iwx^iy \in L$.

The d limits the size of the part that gets pumped.

Using the pumping lemma

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

Example use

Theorem

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

Example use

Theorem

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

For the proof, we assume that L is context-free. Let $r = \max c\ d$.

Example use

Theorem

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

For the proof, we assume that L is context-free. Let $r = \max c\ d$.

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

Example use

Theorem

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

For the proof, we assume that L is context-free. Let $r = \max c\ d$.

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

From the pumping lemma, we learn that we can pump z, and that the part that gets pumped is no longer than d, thus smaller than r.

Example use

Theorem

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

For the proof, we assume that L is context-free. Let $r = \max c\ d$.

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

From the pumping lemma, we learn that we can pump z, and that the part that gets pumped is no longer than d, thus smaller than r.

The part being pumped can thus not contain all of as, bs and cs, but is not empty either, leading to a contradiction.

Exercise

One of these languages is context-free (write the grammar or the PDA) and one is not (prove it using the pumping lemma):

- $L = \{ww \mid w \in \{a, b\}^*\}$
- $L = \{w\, w^{rev} \mid w \in \{a, b\}^*\}$
  - where x^{rev} is the **reverse** of the word x.