COMP20121 The Implementation and Power of Computer Languages

‘Power’ Part

http://www.cs.man.ac.uk/~petera/2121/index.html .

Peter Aczel

room: CS2.52, tel: 56155

email: petera@cs.man.ac.uk

Department of Computer Science, University of Manchester

COMP20121, ‘power’ part: section 2

lecture 5: Pushdown Automata

LECTURE SIX

From Grammars to PDAs

From grammars to PDAs

Given a grammar, create a PDA:

From grammars to PDAs

Given a grammar, create a PDA:

- Three states, 0 (start state), 1 and 2 (accepting state), with transitions

From grammars to PDAs

Given a grammar, create a PDA:

- Three states, 0 (start state), 1 and 2 (accepting state), with transitions
- $0 \xrightarrow{(\,,\,)\mapsto \mathsf{push}(S)} 1$,
- $1 \xrightarrow{(\mathsf{EOF},\mathsf{EOS})\mapsto} 2$,

From grammars to PDAs

Given a grammar, create a PDA:

- Three states, 0 (start state), 1 and 2 (accepting state), with transitions
- $0 \xrightarrow{(\,,)\mapsto \mathsf{push}(S)} 1$,
- $1 \xrightarrow{(\mathsf{EOF},\mathsf{EOS})\mapsto} 2$,
- $1 \xrightarrow{(\ ,R)\mapsto \mathsf{pop};\mathsf{push}(X_n)\cdots\mathsf{push}(X_1)} 1$ for every production $R \to X_1 X_2 \cdots X_n$.

From grammars to PDAs

Given a grammar, create a PDA:

- Three states, 0 (start state), 1 and 2 (accepting state), with transitions
- $0 \xrightarrow{(\,,)\mapsto \mathsf{push}(S)} 1$,
- $1 \xrightarrow{(\mathsf{EOF},\mathsf{EOS})\mapsto} 2$,
- $1 \xrightarrow{(\ ,R)\mapsto \mathsf{pop};\mathsf{push}(X_n)\cdots\mathsf{push}(X_1)} 1$ for every production $R \to X_1 X_2 \cdots X_n$.
- $1 \xrightarrow{(x,x)\mapsto \mathsf{pop};\mathsf{advance}} 1$, for each $x \in \Sigma$.

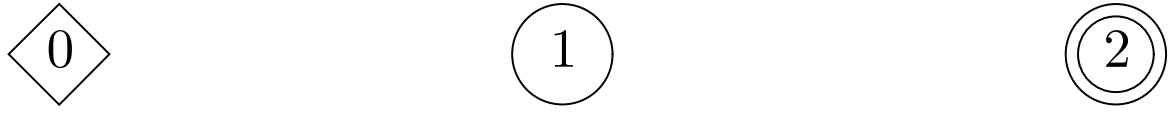
Example: From grammar to PDA

Consider the grammar $S \longrightarrow aSb|\epsilon;$

Example: From grammar to PDA

Consider the grammar $S \longrightarrow aSb|\epsilon;$

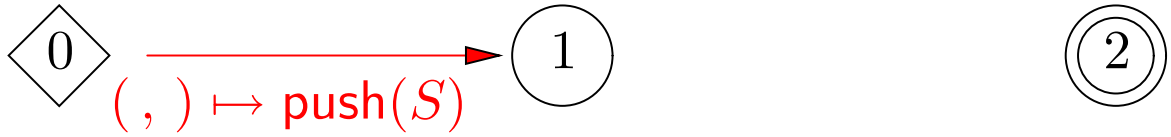
Build a PDA: Three states



Example: From grammar to PDA

Consider the grammar $S \longrightarrow aSb|\epsilon;$

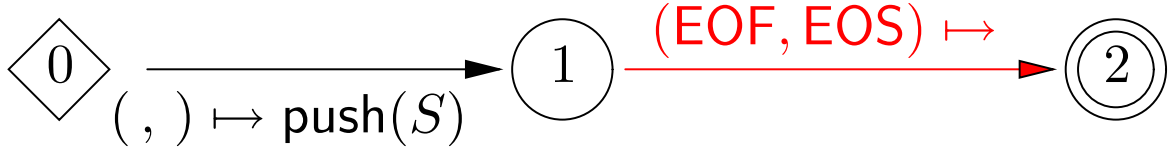
Build a PDA: Getting started.



Example: From grammar to PDA

Consider the grammar $S \longrightarrow aSb|\epsilon;$

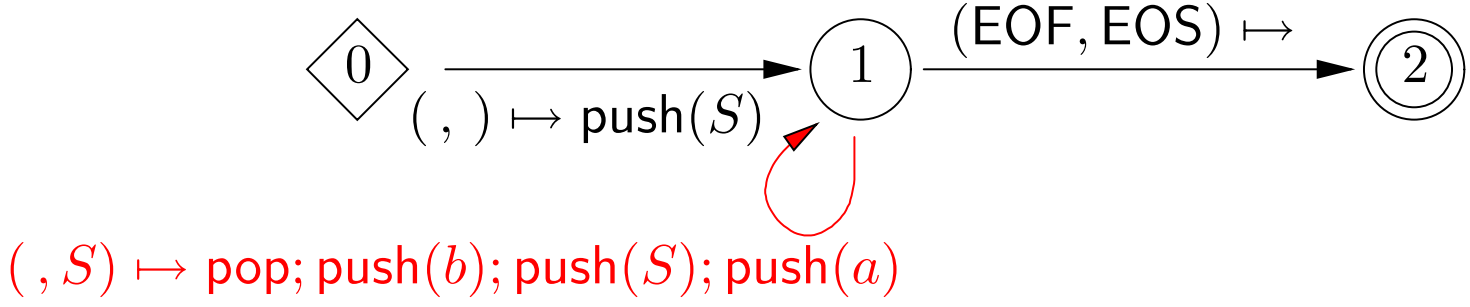
Build a PDA: When finished.



Example: From grammar to PDA

Consider the grammar $S \longrightarrow aSb|\epsilon;$

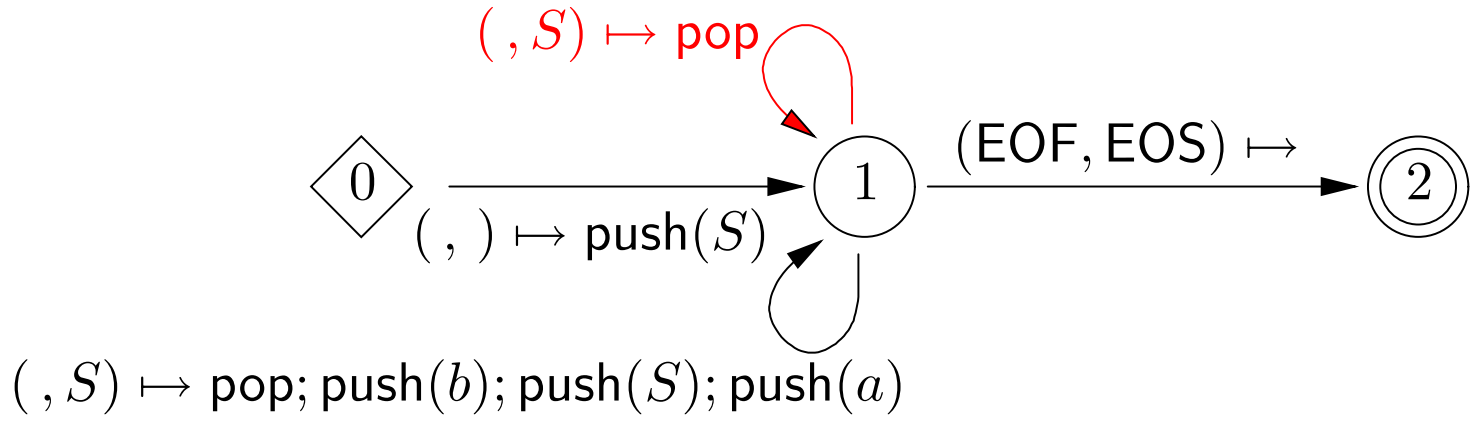
Build a PDA: $S \longrightarrow aSb$.



Example: From grammar to PDA

Consider the grammar $S \longrightarrow aSb|\epsilon;$

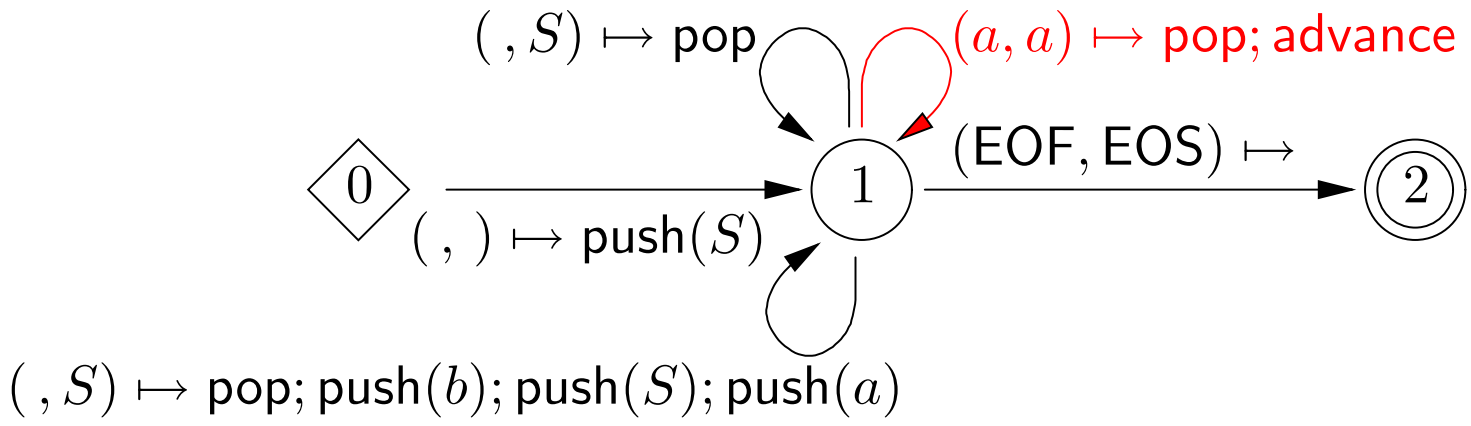
Build a PDA: $S \longrightarrow \epsilon$.



Example: From grammar to PDA

Consider the grammar $S \longrightarrow aSb|\epsilon;$

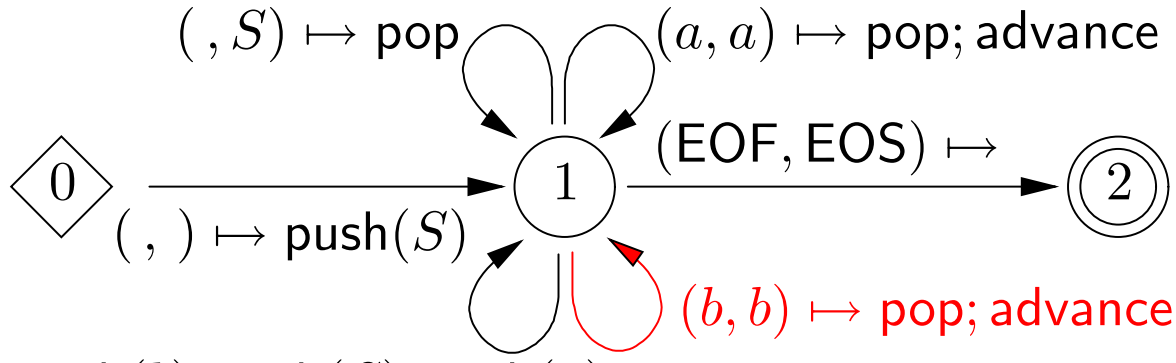
Build a PDA: Recognize a.



Example: From grammar to PDA

Consider the grammar $S \longrightarrow aSb|\epsilon;$

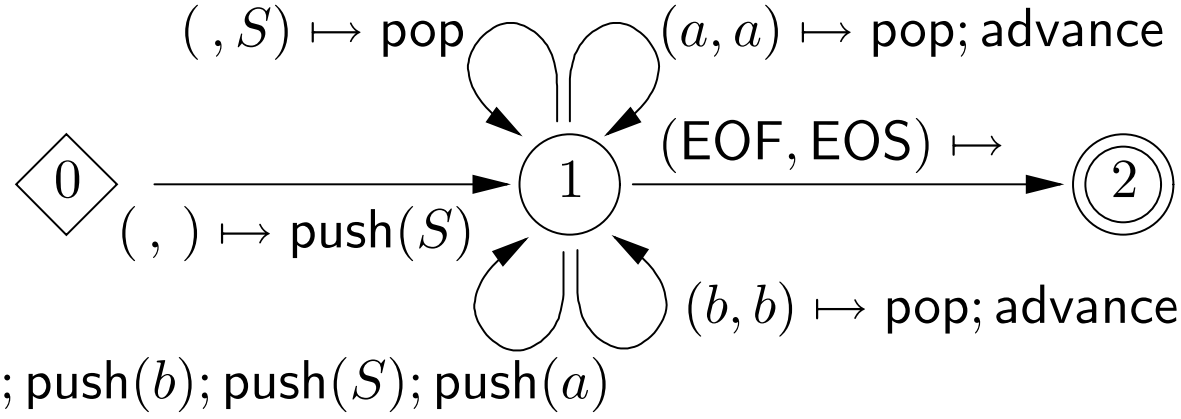
Build a PDA: Recognize b.



Example: From grammar to PDA

Consider the grammar $S \longrightarrow aSb|\epsilon;$

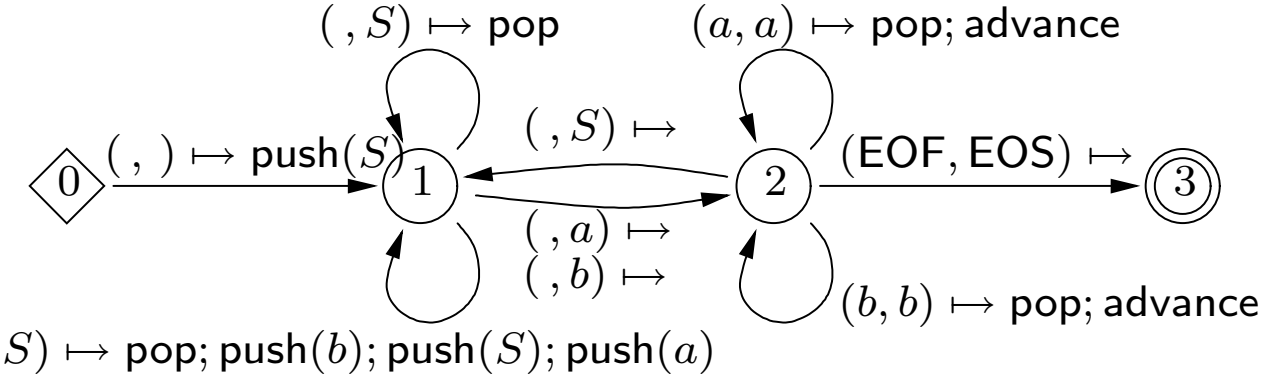
Finished PDA



Example: From grammar to PDA

Consider the grammar $S \to aSb|\epsilon;$

which (with exception of the empty word) accepts the same language as the PDA



From PDAs to grammars

- Given a (non-deterministic) PDA, it is possible to devise a context-free grammar such that the language generated by the grammar is the language accepted by the automaton.

From PDAs to grammars

- Given a (non-deterministic) PDA, it is possible to devise a context-free grammar such that the language generated by the grammar is the language accepted by the automaton.
- We will not prove this statement here. See any of the recommended books.

Context-free languages and PDAs

The following result connects context-free languages and PDAs.

Theorem 2.1 *A language is context-free if and only if there is a non-deterministic pushdown automaton that accepts it.*

Determinism *versus* non-determinism

- Non-deterministic PDAs are more powerful than deterministic PDAs.

Determinism *versus* non-determinism

- Non-deterministic PDAs are more powerful than deterministic PDAs.
- There are context-free grammars for which there is no deterministic pushdown automaton. See Exercise 19 for an example.

Determinism *versus* non-determinism

- Non-deterministic PDAs are more powerful than deterministic PDAs.
- There are context-free grammars for which there is no deterministic pushdown automaton. See Exercise 19 for an example.

(This example is not examinable.)

Long words in context-free languages

If α is a 'long enough' word then in the parse tree for this word, at least one of the non-terminal symbols, say R, will appear twice.

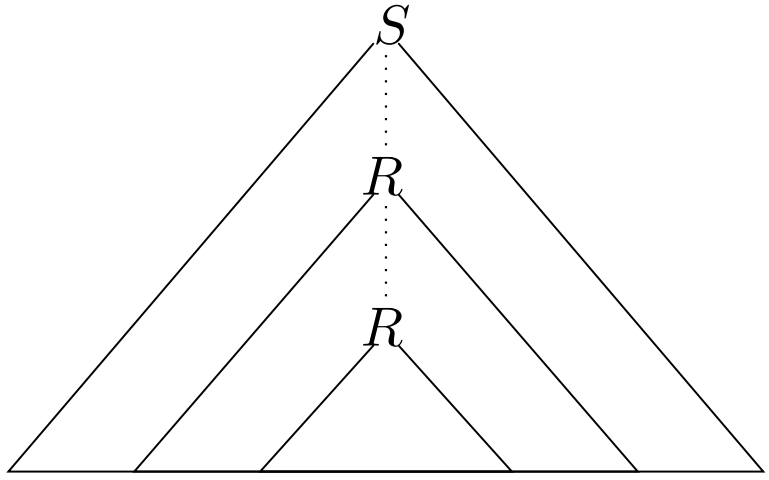
Long words in context-free languages

If α is a 'long enough' word then in the parse tree for this word, at least one of the non-terminal symbols, say R, will appear twice.

It is not easy to define what 'long enough' means this time.

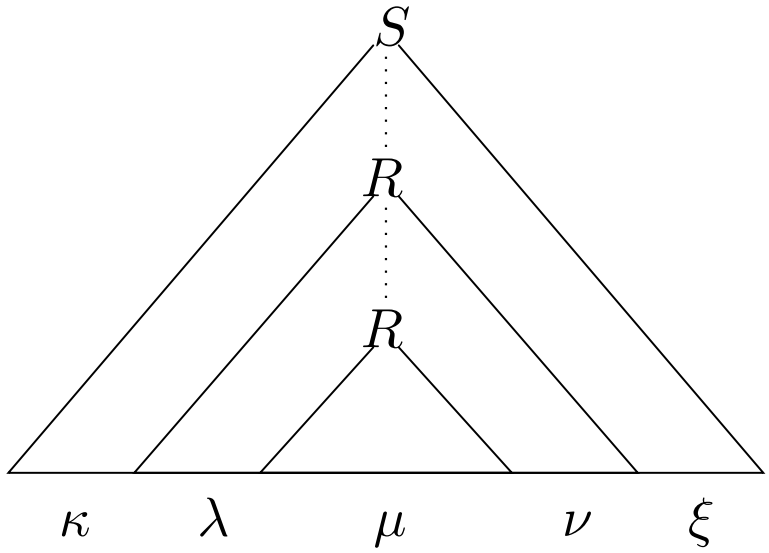
Long words in context-free languages

Parse tree for α:



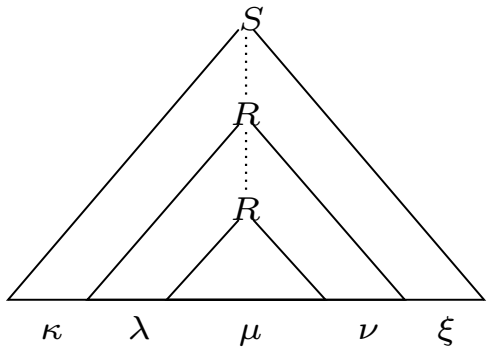
Long words in context-free languages

Breaking up α:



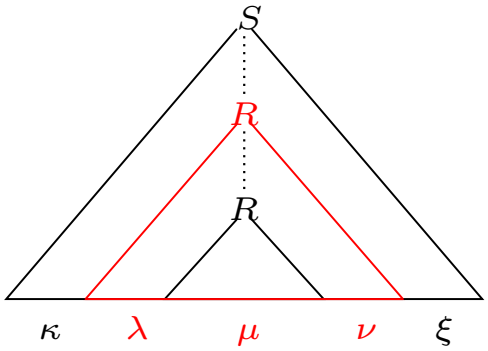
Leaving out parts of α

Given a parse tree for α, with α broken up as before,



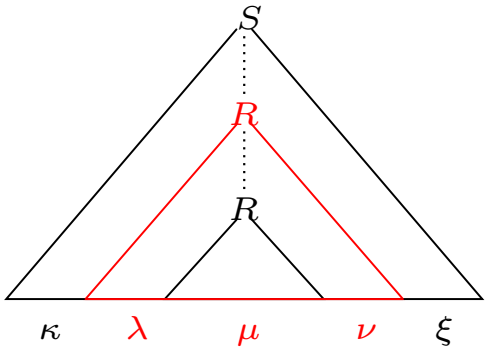
Leaving out parts of α

we can replace the big parse tree at R

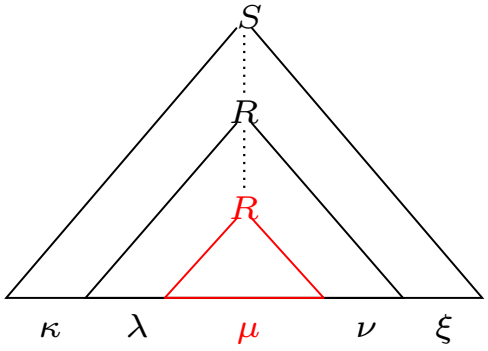


Leaving out parts of α

we can replace the big parse tree at R

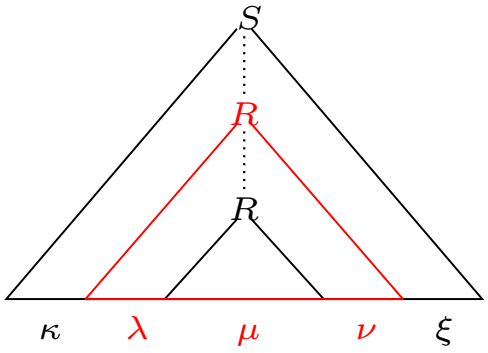


by the small parse tree at R

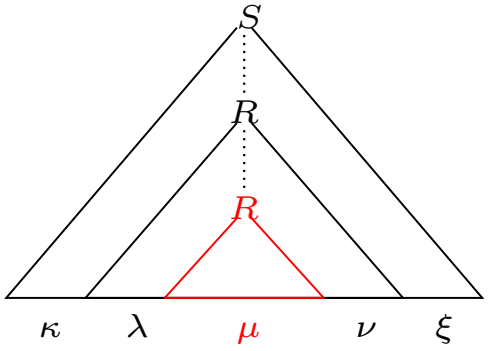


Leaving out parts of α

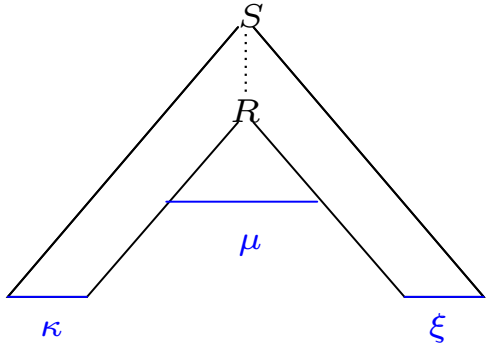
we can replace the big parse tree at R



by the small parse tree at R

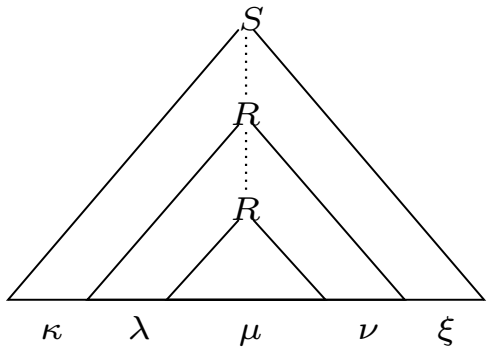


to get the parse tree for $\kappa\mu\xi$



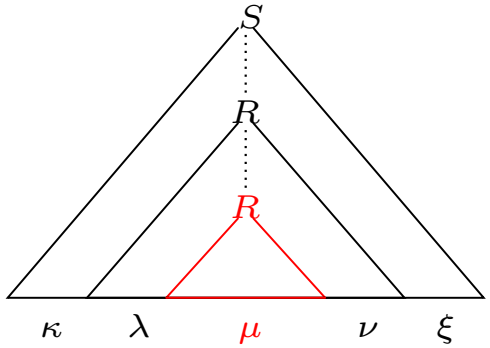
Repeating parts of α

Given a parse tree for α, with α broken up as before,



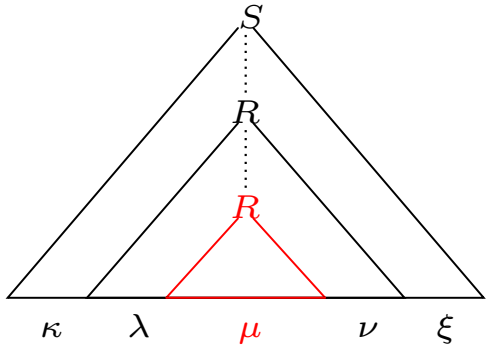
Repeating parts of α

we can replace the small parse tree at R

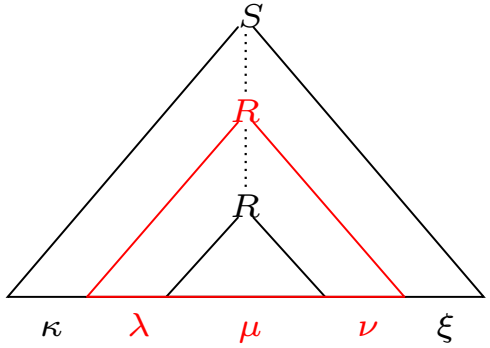


Repeating parts of α

we can replace the small parse tree at R

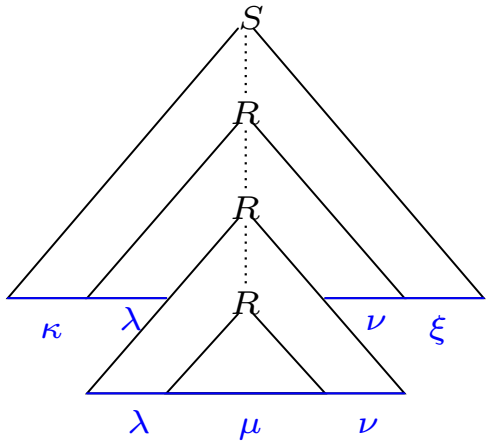


by the big parse tree at R



Repeating parts of α

to get the parse tree for $\kappa\lambda\lambda\mu\nu\nu\xi$



The Pumping Lemma II

We can formalize this observation:

Lemma 2.2 *If L is a context-free language then there is a number n such that any word α of L that has length at least n can be divided into five pieces*

The Pumping Lemma II

$\alpha = \kappa\lambda\mu\nu\xi$ *such that*

- *the length of* $\lambda\nu$ *is greater than* 0*;*
- *the length of* $\lambda\mu\nu$ *is at most* n*;*
- *all words of the form* $\kappa\lambda^k\mu\nu^k\xi$*, where* $k \in \mathbb{N}$*, belong to* L*.*

The Pumping Lemma II

$\alpha = \kappa\lambda\mu\nu\xi$ such that

- *the length of $\lambda\nu$ is greater than 0;*
- *the length of $\lambda\mu\nu$ is at most n;*
- *all words of the form $\kappa\lambda^k\mu\nu^k\xi$, where $k \in \mathbb{N}$, belong to L.*

Note that if L is finite, it may not have any words of length n or longer!

Applying Pumping: An example

- The language $L = \{a^i b^i c^i \mid i \in N\}$ is

not context-free.

Applying Pumping: An example

- The language $L = \{a^i b^i c^i \mid i \in N\}$ is

not context-free.

- To see this, suppose it were!

Choose n by the Pumping Lemma.

Let $\alpha = a^n b^n c^n$.
Write $\alpha = \kappa\lambda\mu\nu\xi$ as in the lemma.

Applying Pumping: An example

Then one of λ, ν is non-empty.

Case 1: Each only contain one kind of character.

Then $\kappa\mu\xi \notin L$.

Applying Pumping: An example

Then one of λ, ν is non-empty.

Case 1: Each only contain one kind of character.

Then $\kappa\mu\xi \notin L$.

Case 2: One of λ, ν contains more than one kind of character.

Then $\kappa\lambda^2\mu\nu^2\xi \notin L$.

Section 2 summary

- Context-free languages are generated by context-free grammars.

Section 2 summary

- Context-free languages are generated by context-free grammars.
- Context-free languages are more powerful than regular ones.

Section 2 summary

- Context-free languages are generated by context-free grammars.
- Context-free languages are more powerful than regular ones.
- Context-free languages correspond precisely to those accepted by non-deterministic pushdown automata.

Section 2 summary (ctd)

- Deterministic PDAs are less powerful than (non-deterministic) PDAs.

Section 2 summary (ctd)

- Deterministic PDAs are less powerful than (non-deterministic) PDAs.
- There is a Pumping Lemma for context-free languages.

Section 2 summary (ctd)

- Deterministic PDAs are less powerful than (non-deterministic) PDAs.
- There is a Pumping Lemma for context-free languages.
- Context-free languages are sufficient to parse programming languages but

Section 2 summary (ctd)

- Deterministic PDAs are less powerful than (non-deterministic) PDAs.
- There is a Pumping Lemma for context-free languages.
- Context-free languages are sufficient to parse programming languages but
- a compiler needs a proper random-access memory.