Automata Theory :: LL Parsing

Jörg Endrullis

Vrije Universiteit Amsterdam

Top-down Parsing

Top-down parsing tries to derive the input word from the starting variable S.

Top-down Parsing

Top-down parsing tries to derive the input word from the starting variable S.

Simple leftmost strategy:

- Always expand the leftmost variable A.
  (Replace A by u if there is a rule $A \to u$.)
- Backtrack when a mismatch with the input string is found.
  (Then try another rule.)

Top-down Parsing

Top-down parsing tries to derive the input word from the starting variable S.

Simple leftmost strategy:

- Always expand the leftmost variable A.
  (Replace A by u if there is a rule $A \rightarrow u$.)
- Backtrack when a mismatch with the input string is found.
  (Then try another rule.)

Disadvantage: backtracking is expensive and difficult.

LL Parsing

LL parsing

Parsing **top-down** with a **leftmost** strategy.

Backtracking is not allowed.

LL parsing does not work for every context-free grammar.

LL Parsing

LL parsing

Parsing **top-down** with a **leftmost** strategy.

Backtracking is not allowed.

LL parsing does not work for every context-free grammar.

Starting point is a context-free grammar $G = (V, T, S, P)$:

- without useless variables
- λ-productions and unit productions are allowed (elimination often increases the size of the grammar)

LL Parsing

LL parsing

Parsing **top-down** with a **leftmost** strategy.

Backtracking is not allowed.

LL parsing does not work for every context-free grammar.

Starting point is a context-free grammar $G = (V, T, S, P)$:

- without useless variables
- λ-productions and unit productions are allowed
  (elimination often increases the size of the grammar)

Steps of LL parsing:

- Construct sets $\text{First}(A)$ and $\text{Follow}(A)$ for every variable A.
- Construct a parsing table.
- Parse the input word using the parsing table.

Useless Variables

Removal of Useless Variables

A variable A is **useless** for a context-free grammar if there exists no derivation of the form

$$S \Rightarrow^* uAv \Rightarrow^+ w \qquad \text{with } w \in T^*.$$

Removal of Useless Variables

A variable A is **useless** for a context-free grammar if there exists no derivation of the form

$$S \Rightarrow^* uAv \Rightarrow^+ w \qquad \text{with } w \in T^*.$$

Removing production rules that contain a useless variable from a grammar does not change the generated language.

Removal of Useless Variables

A variable A is **useless** for a context-free grammar if there exists no derivation of the form

$$S \Rightarrow^* uAv \Rightarrow^+ w \qquad \text{with } w \in T^*.$$

Removing production rules that contain a useless variable from a grammar does not change the generated language.

$$S \to aSb \mid BC \mid \lambda \qquad A \to Sb \qquad B \to a \qquad C \to C$$

Which variables are useless?

Removal of Useless Variables

A variable A is **useless** for a context-free grammar if there exists no derivation of the form

$$S \Rightarrow^* uAv \Rightarrow^+ w \qquad \text{with } w \in T^*.$$

Removing production rules that contain a useless variable from a grammar does not change the generated language.

$$S \to aSb \mid BC \mid \lambda \qquad A \to Sb \qquad B \to a \qquad C \to C$$

Which variables are useless?

- A because there is no derivation $S \Rightarrow^* uAv$

Removal of Useless Variables

A variable A is **useless** for a context-free grammar if there exists no derivation of the form

$$S \Rightarrow^* uAv \Rightarrow^+ w \qquad \text{with } w \in T^*.$$

Removing production rules that contain a useless variable from a grammar does not change the generated language.

$$S \to aSb \mid BC \mid \lambda \qquad A \to Sb \qquad B \to a \qquad C \to C$$

Which variables are useless?

- A because there is no derivation $S \Rightarrow^* uAv$
- C because there is no derivation $C \Rightarrow^* w$ with $w \in T^*$

Removal of Useless Variables

A variable A is **useless** for a context-free grammar if there exists no derivation of the form

$$S \Rightarrow^* uAv \Rightarrow^+ w \qquad \text{with } w \in T^*.$$

Removing production rules that contain a useless variable from a grammar does not change the generated language.

$$S \to aSb \mid BC \mid \lambda \qquad A \to Sb \qquad B \to a \qquad C \to C$$

Which variables are useless?

- A because there is no derivation $S \Rightarrow^* uAv$
- C because there is no derivation $C \Rightarrow^* w$ with $w \in T^*$
- B because B can be reached only together with C

Removal of Useless Variables

A variable A is **useless** for a context-free grammar if there exists no derivation of the form

$$S \Rightarrow^* uAv \Rightarrow^+ w \qquad \text{with } w \in T^*.$$

Removing production rules that contain a useless variable from a grammar does not change the generated language.

$$S \to aSb \mid BC \mid \lambda \qquad A \to Sb \qquad B \to a \qquad C \to C$$

Which variables are useless?

- A because there is no derivation $S \Rightarrow^* uAv$
- C because there is no derivation $C \Rightarrow^* w$ with $w \in T^*$
- B because B can be reached only together with C

The resulting grammar is $S \to aSb \mid \lambda$.

Removal of Useless Variables

Question

How to determine useless variables of a context-free grammar?

Removal of Useless Variables

Question

How to determine useless variables of a context-free grammar?

Construction

A variable A is called **productive** if $A \Rightarrow^+ w$ with $w \in T^*$.

Removal of Useless Variables

Question

How to determine useless variables of a context-free grammar?

Construction

A variable A is called **productive** if $A \Rightarrow^{+} w$ with $w \in T^*$.

We determine all productive variables:

- If $A \rightarrow y$ is a rule and all variables in y are productive, then A is productive.

Removal of Useless Variables

Question

How to determine useless variables of a context-free grammar?

Construction

A variable A is called **productive** if $A \Rightarrow^+ w$ with $w \in T^*$.

We determine all productive variables:

- If $A \rightarrow y$ is a rule and all variables in y are productive, then A is productive.

Remove all rules that contain a non-productive variable.

Removal of Useless Variables

Question

How to determine useless variables of a context-free grammar?

Construction

A variable A is called **productive** if $A \Rightarrow^+ w$ with $w \in T^*$.

We determine all productive variables:

- If $A \to y$ is a rule and all variables in y are productive, then A is productive.

Remove all rules that contain a non-productive variable.

We determine all **reachable** variables as follows:

- S is reachable.
- If $A \to y$ and A is reachable, then so are all variables in y.

Removal of Useless Variables

Question

How to determine useless variables of a context-free grammar?

Construction

A variable A is called **productive** if $A \Rightarrow^{+} w$ with $w \in T^*$.

We determine all productive variables:

- If $A \rightarrow y$ is a rule and all variables in y are productive, then A is productive.

Remove all rules that contain a non-productive variable.

We determine all **reachable** variables as follows:

- S is reachable.
- If $A \rightarrow y$ and A is reachable, then so are all variables in y.

Remove all rules that contain a non-reachable variable.

Removal of Useless Variables

Question

How to determine useless variables of a context-free grammar?

Construction

A variable A is called **productive** if $A \Rightarrow^+ w$ with $w \in T^*$.

We determine all productive variables:

- If $A \rightarrow y$ is a rule and all variables in y are productive, then A is productive.

Remove all rules that contain a non-productive variable.

We determine all **reachable** variables as follows:

- S is reachable.
- If $A \rightarrow y$ and A is reachable, then so are all variables in y.

Remove all rules that contain a non-reachable variable.

A variable is useless if it is not in one of the remaining rules.

Removal of Useless Variables

$$S \to aSb \mid BC \mid \lambda \qquad A \to Sb \qquad B \to a \qquad C \to C$$

Removal of Useless Variables

$$S \to aSb \mid BC \mid \lambda \qquad A \to Sb \qquad B \to a \qquad C \to C$$

Which variables are non-productive?

Removal of Useless Variables

$$S \to aSb \mid BC \mid \lambda \qquad A \to Sb \qquad B \to a \qquad C \to C$$

Which variables are non-productive?

- C is not productive

Removal of Useless Variables

$$S \to aSb \mid BC \mid \lambda \qquad A \to Sb \qquad B \to a \qquad C \to C$$

Which variables are non-productive?

- C is not productive

We remove all rules containing non-productive variables:

$$S \to aSb \mid \lambda \qquad A \to Sb \qquad B \to a$$

Removal of Useless Variables

$$S \to aSb \mid BC \mid \lambda \qquad A \to Sb \qquad B \to a \qquad C \to C$$

Which variables are non-productive?

- C is not productive

We remove all rules containing non-productive variables:

$$S \to aSb \mid \lambda \qquad A \to Sb \qquad B \to a$$

Which variables are reachable from S?

Removal of Useless Variables

$$S \rightarrow aSb \mid BC \mid \lambda \qquad A \rightarrow Sb \qquad B \rightarrow a \qquad C \rightarrow C$$

Which variables are non-productive?

- C is not productive

We remove all rules containing non-productive variables:

$$S \rightarrow aSb \mid \lambda \qquad A \rightarrow Sb \qquad B \rightarrow a$$

Which variables are reachable from S?

- only S is reachable

Removal of Useless Variables

$$S \to aSb \mid BC \mid \lambda \qquad A \to Sb \qquad B \to a \qquad C \to C$$

Which variables are non-productive?

- C is not productive

We remove all rules containing non-productive variables:

$$S \to aSb \mid \lambda \qquad A \to Sb \qquad B \to a$$

Which variables are reachable from S?

- only S is reachable

We remove all rules containing non-reachable variables:

$$S \to aSb \mid \lambda$$

Removal of Useless Variables

$$S \to aSb \mid BC \mid \lambda \qquad A \to Sb \qquad B \to a \qquad C \to C$$

Which variables are non-productive?

- C is not productive

We remove all rules containing non-productive variables:

$$S \to aSb \mid \lambda \qquad A \to Sb \qquad B \to a$$

Which variables are reachable from S?

- only S is reachable

We remove all rules containing non-reachable variables:

$$S \to aSb \mid \lambda$$

Hence only S is useful, the variables A, B, C are not useful.

$\text{First}(A)$

First(A)

We consider the first terminal letters derivable from a word:

$$\text{First}(w) = \{\, a \in T \mid w \Rightarrow^* a \ldots \} \cup \{\lambda \mid w \Rightarrow^* \lambda\}$$

First(A)

We consider the first terminal letters derivable from a word:

$$\text{First}(w) = \{\, a \in T \mid w \Rightarrow^* a \ldots \} \cup \{\, \lambda \mid w \Rightarrow^* \lambda \,\}$$

Algorithm

Let PreFirst(w) be the smallest set such that:

- $w \in \text{PreFirst}(w)$
- $a \in \text{PreFirst}(w)$ if $av \in \text{PreFirst}(w)$
- $B \in \text{PreFirst}(w)$ if $Bv \in \text{PreFirst}(w)$
- $v \in \text{PreFirst}(w)$ if $Bv \in \text{PreFirst}(w)$ and B erasable
- $v \in \text{PreFirst}(w)$ for every $A \in \text{PreFirst}(w)$ and rule $A \to v$

Then First(w) consists of

- all terminal letters $a \in T$ such that $a \in \text{PreFirst}(w)$, and
- λ if $w = A_1 A_2 \ldots A_n$ for erasable variables $A_1, \ldots, A_n$.

Exercise

$$S \to AAc \qquad A \to Ba \mid \lambda \qquad B \to Ab \mid d$$

Exercise

$$S \to AAc \qquad A \to Ba \mid \lambda \qquad B \to Ab \mid d$$

The erasable variables ($V \Rightarrow^{+} \lambda$) are:

Exercise

$$S \to AAc \qquad A \to Ba \mid \lambda \qquad B \to Ab \mid d$$

The erasable variables ($V \Rightarrow^{+} \lambda$) are: A .

Exercise

$$S \rightarrow AAc \qquad A \rightarrow Ba \mid \lambda \qquad B \rightarrow Ab \mid d$$

The erasable variables ($V \Rightarrow^+ \lambda$) are: A .

We determine PreFirst(A), PreFirst(B) and PreFirst(S):

Exercise

$$S \to AAc \qquad A \to Ba \mid \lambda \qquad B \to Ab \mid d$$

The erasable variables ($V \Rightarrow^+ \lambda$) are: A .

We determine PreFirst(A), PreFirst(B) and PreFirst(S):

$\text{PreFirst}(A) = \{ \qquad \}$

$\text{PreFirst}(B) = \{ \qquad \}$

$\text{PreFirst}(S) = \{ \qquad \}$

Exercise

$$S \rightarrow AAc \qquad A \rightarrow Ba \mid \lambda \qquad B \rightarrow Ab \mid d$$

The erasable variables ($V \Rightarrow^{+} \lambda$) are: A .

We determine PreFirst(A), PreFirst(B) and PreFirst(S):

PreFirst$(A) = \{\, A \qquad\qquad\qquad\qquad \}$

PreFirst$(B) = \{\qquad\qquad\qquad\qquad \}$

PreFirst$(S) = \{\qquad\qquad\qquad\qquad \}$

Exercise

$$S \to AAc \qquad A \to Ba \mid \lambda \qquad B \to Ab \mid d$$

The erasable variables ($V \Rightarrow^+ \lambda$) are: A .

We determine PreFirst(A), PreFirst(B) and PreFirst(S):

$$\text{PreFirst}(A) = \{\, A,\ \underbrace{Ba}_{\text{from } A} \qquad \}$$

$$\text{PreFirst}(B) = \{ \qquad \}$$

$$\text{PreFirst}(S) = \{ \qquad \}$$

Exercise

$$S \to AAc \qquad A \to Ba \mid \lambda \qquad B \to Ab \mid d$$

The erasable variables ($V \Rightarrow^+ \lambda$) are: A .

We determine PreFirst(A), PreFirst(B) and PreFirst(S):

$$\text{PreFirst}(A) = \{\, A, \underbrace{Ba}_{\text{from } A}, \underbrace{\lambda}_{\text{from } A} \qquad \}$$

$$\text{PreFirst}(B) = \{ \qquad \}$$

$$\text{PreFirst}(S) = \{ \qquad \}$$

Exercise

$$S \to AAc \qquad A \to Ba \mid \lambda \qquad B \to Ab \mid d$$

The erasable variables ($V \Rightarrow^+ \lambda$) are: A .

We determine PreFirst(A), PreFirst(B) and PreFirst(S):

$$\text{PreFirst}(A) = \{\, A, \underbrace{Ba}_{\text{from } A}, \underbrace{\lambda}_{\text{from } A}, \underbrace{B}_{\text{from } Ba} \qquad \}$$

$$\text{PreFirst}(B) = \{ \qquad \}$$

$$\text{PreFirst}(S) = \{ \qquad \}$$

Exercise

$$S \to AAc \qquad A \to Ba \mid \lambda \qquad B \to Ab \mid d$$

The erasable variables ($V \Rightarrow^+ \lambda$) are: A .

We determine PreFirst(A), PreFirst(B) and PreFirst(S):

$$\text{PreFirst}(A) = \{\, A,\ \underbrace{Ba}_{\text{from } A},\ \underbrace{\lambda}_{\text{from } A},\ \underbrace{B}_{\text{from } Ba},\ \underbrace{Ab}_{\text{from } B} \qquad \}$$

$$\text{PreFirst}(B) = \{ \qquad \}$$

$$\text{PreFirst}(S) = \{ \qquad \}$$

Exercise

$$S \to AAc \qquad A \to Ba \mid \lambda \qquad B \to Ab \mid d$$

The erasable variables ($V \Rightarrow^+ \lambda$) are: A .

We determine PreFirst(A), PreFirst(B) and PreFirst(S):

$$\text{PreFirst}(A) = \{\, A, \underbrace{Ba}_{\text{from } A}, \underbrace{\lambda}_{\text{from } A}, \underbrace{B}_{\text{from } Ba}, \underbrace{Ab}_{\text{from } B}, \underbrace{d}_{\text{from } B} \quad \}$$

$$\text{PreFirst}(B) = \{ \quad \}$$

$$\text{PreFirst}(S) = \{ \quad \}$$

Exercise

$$S \to AAc \qquad A \to Ba \mid \lambda \qquad B \to Ab \mid d$$

The erasable variables ($V \Rightarrow^+ \lambda$) are: A .

We determine PreFirst(A), PreFirst(B) and PreFirst(S):

$$\text{PreFirst}(A) = \{ A, \underbrace{Ba}_{\text{from } A}, \underbrace{\lambda}_{\text{from } A}, \underbrace{B}_{\text{from } Ba}, \underbrace{Ab}_{\text{from } B}, \underbrace{d}_{\text{from } B}, \underbrace{b}_{\text{from } Ab} \}$$

$$\text{PreFirst}(B) = \{ \qquad \}$$

$$\text{PreFirst}(S) = \{ \qquad \}$$

Exercise

$$S \rightarrow AAc \qquad A \rightarrow Ba \mid \lambda \qquad B \rightarrow Ab \mid d$$

The erasable variables ($V \Rightarrow^{+} \lambda$) are: A .

We determine PreFirst(A), PreFirst(B) and PreFirst(S):

$$\text{PreFirst}(A) = \{\, A,\ \underbrace{Ba}_{\text{from } A},\ \underbrace{\lambda}_{\text{from } A},\ \underbrace{B}_{\text{from } Ba},\ \underbrace{Ab}_{\text{from } B},\ \underbrace{d}_{\text{from } B},\ \underbrace{b}_{\text{from } Ab} \,\}$$

$$\text{PreFirst}(B) = \{\, B \qquad\qquad \}$$

$$\text{PreFirst}(S) = \{ \qquad\qquad \}$$

Exercise

$$S \to AAc \qquad A \to Ba \mid \lambda \qquad B \to Ab \mid d$$

The erasable variables ($V \Rightarrow^+ \lambda$) are: A .

We determine PreFirst(A), PreFirst(B) and PreFirst(S):

$$\text{PreFirst}(A) = \{\, A, \underbrace{Ba}_{\text{from } A}, \underbrace{\lambda}_{\text{from } A}, \underbrace{B}_{\text{from } Ba}, \underbrace{Ab}_{\text{from } B}, \underbrace{d}_{\text{from } B}, \underbrace{b}_{\text{from } Ab} \,\}$$

$$\text{PreFirst}(B) = \{\, B, \underbrace{Ab}_{\text{from } B} \qquad \}$$

$$\text{PreFirst}(S) = \{ \qquad \}$$

Exercise

$$S \to AAc \qquad A \to Ba \mid \lambda \qquad B \to Ab \mid d$$

The erasable variables ($V \Rightarrow^+ \lambda$) are: A .

We determine PreFirst(A), PreFirst(B) and PreFirst(S):

$$\text{PreFirst}(A) = \{\, A, \underbrace{Ba}_{\text{from } A}, \underbrace{\lambda}_{\text{from } A}, \underbrace{B}_{\text{from } Ba}, \underbrace{Ab}_{\text{from } B}, \underbrace{d}_{\text{from } B}, \underbrace{b}_{\text{from } Ab} \,\}$$

$$\text{PreFirst}(B) = \{\, B, \underbrace{Ab}_{\text{from } B}, \underbrace{d}_{\text{from } B} \,\}$$

$$\text{PreFirst}(S) = \{\quad\}$$

Exercise

$$S \to AAc \qquad A \to Ba \mid \lambda \qquad B \to Ab \mid d$$

The erasable variables ($V \Rightarrow^+ \lambda$) are: A .

We determine PreFirst(A), PreFirst(B) and PreFirst(S):

$$\text{PreFirst}(A) = \{\, A,\ \underbrace{Ba}_{\text{from } A},\ \underbrace{\lambda}_{\text{from } A},\ \underbrace{B}_{\text{from } Ba},\ \underbrace{Ab}_{\text{from } B},\ \underbrace{d}_{\text{from } B},\ \underbrace{b}_{\text{from } Ab} \,\}$$

$$\text{PreFirst}(B) = \{\, B,\ \underbrace{Ab}_{\text{from } B},\ \underbrace{d}_{\text{from } B},\ \underbrace{b}_{\text{from } Ab} \,\}$$

$$\text{PreFirst}(S) = \{ \qquad \}$$

Exercise

$$S \to AAc \qquad A \to Ba \mid \lambda \qquad B \to Ab \mid d$$

The erasable variables ($V \Rightarrow^{+} \lambda$) are: A.

We determine PreFirst(A), PreFirst(B) and PreFirst(S):

$$\text{PreFirst}(A) = \{\, A, \underbrace{Ba}_{\text{from } A}, \underbrace{\lambda}_{\text{from } A}, \underbrace{B}_{\text{from } Ba}, \underbrace{Ab}_{\text{from } B}, \underbrace{d}_{\text{from } B}, \underbrace{b}_{\text{from } Ab} \,\}$$

$$\text{PreFirst}(B) = \{\, B, \underbrace{Ab}_{\text{from } B}, \underbrace{d}_{\text{from } B}, \underbrace{b}_{\text{from } Ab}, \underbrace{A}_{\text{from } Ab} \,\}$$

$$\text{PreFirst}(S) = \{ \qquad \}$$

Exercise

$$S \to AAc \qquad A \to Ba \mid \lambda \qquad B \to Ab \mid d$$

The erasable variables ($V \Rightarrow^+ \lambda$) are: A .

We determine PreFirst(A), PreFirst(B) and PreFirst(S):

$$\text{PreFirst}(A) = \{\, A, \underbrace{Ba}_{\text{from } A}, \underbrace{\lambda}_{\text{from } A}, \underbrace{B}_{\text{from } Ba}, \underbrace{Ab}_{\text{from } B}, \underbrace{d}_{\text{from } B}, \underbrace{b}_{\text{from } Ab} \,\}$$

$$\text{PreFirst}(B) = \{\, B, \underbrace{Ab}_{\text{from } B}, \underbrace{d}_{\text{from } B}, \underbrace{b}_{\text{from } Ab}, \underbrace{A}_{\text{from } Ab} \,\} \cup \text{PreFirst}(A)$$

$$\text{PreFirst}(S) = \{ \qquad \}$$

Exercise

$$S \to AAc \qquad A \to Ba \mid \lambda \qquad B \to Ab \mid d$$

The erasable variables ($V \Rightarrow^{+} \lambda$) are: A .

We determine PreFirst(A), PreFirst(B) and PreFirst(S):

$$\text{PreFirst}(A) = \{ A, \underbrace{Ba}_{\text{from } A}, \underbrace{\lambda}_{\text{from } A}, \underbrace{B}_{\text{from } Ba}, \underbrace{Ab}_{\text{from } B}, \underbrace{d}_{\text{from } B}, \underbrace{b}_{\text{from } Ab} \}$$

$$\text{PreFirst}(B) = \{ B, \underbrace{Ab}_{\text{from } B}, \underbrace{d}_{\text{from } B}, \underbrace{b}_{\text{from } Ab}, \underbrace{A}_{\text{from } Ab} \} \cup \text{PreFirst}(A)$$

$$= \{ A, Ba, \lambda, B, Ab, d, b \}$$

$$\text{PreFirst}(S) = \{ \qquad \}$$

Exercise

$$S \to AAc \qquad A \to Ba \mid \lambda \qquad B \to Ab \mid d$$

The erasable variables ($V \Rightarrow^+ \lambda$) are: A .

We determine PreFirst(A), PreFirst(B) and PreFirst(S):

$$\text{PreFirst}(A) = \{ A, \underbrace{Ba}_{\text{from } A}, \underbrace{\lambda}_{\text{from } A}, \underbrace{B}_{\text{from } Ba}, \underbrace{Ab}_{\text{from } B}, \underbrace{d}_{\text{from } B}, \underbrace{b}_{\text{from } Ab} \}$$

$$\text{PreFirst}(B) = \{ B, \underbrace{Ab}_{\text{from } B}, \underbrace{d}_{\text{from } B}, \underbrace{b}_{\text{from } Ab}, \underbrace{A}_{\text{from } Ab} \} \cup \text{PreFirst}(A)$$

$$= \{ A, Ba, \lambda, B, Ab, d, b \}$$

$$\text{PreFirst}(S) = \{ S \qquad \}$$

Exercise

$$S \to AAc \qquad A \to Ba \mid \lambda \qquad B \to Ab \mid d$$

The erasable variables ($V \Rightarrow^{+} \lambda$) are: A.

We determine PreFirst(A), PreFirst(B) and PreFirst(S):

$$\text{PreFirst}(A) = \{\, A, \underbrace{Ba}_{\text{from } A}, \underbrace{\lambda}_{\text{from } A}, \underbrace{B}_{\text{from } Ba}, \underbrace{Ab}_{\text{from } B}, \underbrace{d}_{\text{from } B}, \underbrace{b}_{\text{from } Ab} \,\}$$

$$\text{PreFirst}(B) = \{\, B, \underbrace{Ab}_{\text{from } B}, \underbrace{d}_{\text{from } B}, \underbrace{b}_{\text{from } Ab}, \underbrace{A}_{\text{from } Ab} \,\} \cup \text{PreFirst}(A)$$

$$= \{\, A, Ba, \lambda, B, Ab, d, b \,\}$$

$$\text{PreFirst}(S) = \{\, S, \underbrace{AAc}_{\text{from } S} \qquad \}$$

Exercise

$$S \to AAc \qquad A \to Ba \mid \lambda \qquad B \to Ab \mid d$$

The erasable variables ($V \Rightarrow^+ \lambda$) are: A .

We determine PreFirst(A), PreFirst(B) and PreFirst(S):

$$\text{PreFirst}(A) = \{\, A, \underbrace{Ba}_{\text{from } A}, \underbrace{\lambda}_{\text{from } A}, \underbrace{B}_{\text{from } Ba}, \underbrace{Ab}_{\text{from } B}, \underbrace{d}_{\text{from } B}, \underbrace{b}_{\text{from } Ab} \,\}$$

$$\text{PreFirst}(B) = \{\, B, \underbrace{Ab}_{\text{from } B}, \underbrace{d}_{\text{from } B}, \underbrace{b}_{\text{from } Ab}, \underbrace{A}_{\text{from } Ab} \,\} \cup \text{PreFirst}(A)$$

$$= \{\, A, Ba, \lambda, B, Ab, d, b \,\}$$

$$\text{PreFirst}(S) = \{\, S, \underbrace{AAc}_{\text{from } S}, \underbrace{Ac}_{\text{from } AAc} \qquad \}$$

Exercise

$$S \to AAc \qquad A \to Ba \mid \lambda \qquad B \to Ab \mid d$$

The erasable variables ($V \Rightarrow^{+} \lambda$) are: A.

We determine PreFirst(A), PreFirst(B) and PreFirst(S):

$$\text{PreFirst}(A) = \{ A, \underbrace{Ba}_{\text{from } A}, \underbrace{\lambda}_{\text{from } A}, \underbrace{B}_{\text{from } Ba}, \underbrace{Ab}_{\text{from } B}, \underbrace{d}_{\text{from } B}, \underbrace{b}_{\text{from } Ab} \}$$

$$\text{PreFirst}(B) = \{ B, \underbrace{Ab}_{\text{from } B}, \underbrace{d}_{\text{from } B}, \underbrace{b}_{\text{from } Ab}, \underbrace{A}_{\text{from } Ab} \} \cup \text{PreFirst}(A)$$

$$= \{ A, Ba, \lambda, B, Ab, d, b \}$$

$$\text{PreFirst}(S) = \{ S, \underbrace{AAc}_{\text{from } S}, \underbrace{Ac}_{\text{from } AAc}, \underbrace{c}_{\text{from } Ac} \}$$

Exercise

$$S \to AAc \qquad A \to Ba \mid \lambda \qquad B \to Ab \mid d$$

The erasable variables ($V \Rightarrow^+ \lambda$) are: A .

We determine PreFirst(A), PreFirst(B) and PreFirst(S):

$$\text{PreFirst}(A) = \{\, A, \underbrace{Ba}_{\text{from } A}, \underbrace{\lambda}_{\text{from } A}, \underbrace{B}_{\text{from } Ba}, \underbrace{Ab}_{\text{from } B}, \underbrace{d}_{\text{from } B}, \underbrace{b}_{\text{from } Ab} \,\}$$

$$\text{PreFirst}(B) = \{\, B, \underbrace{Ab}_{\text{from } B}, \underbrace{d}_{\text{from } B}, \underbrace{b}_{\text{from } Ab}, \underbrace{A}_{\text{from } Ab} \,\} \cup \text{PreFirst}(A)$$

$$= \{\, A, Ba, \lambda, B, Ab, d, b \,\}$$

$$\text{PreFirst}(S) = \{\, S, \underbrace{AAc}_{\text{from } S}, \underbrace{Ac}_{\text{from } AAc}, \underbrace{c}_{\text{from } Ac}, \underbrace{A}_{\text{from } AAc} \,\}$$

Exercise

$$S \to AAc \qquad A \to Ba \mid \lambda \qquad B \to Ab \mid d$$

The erasable variables ($V \Rightarrow^+ \lambda$) are: A .

We determine PreFirst(A), PreFirst(B) and PreFirst(S):

$$\text{PreFirst}(A) = \{\, A, \underbrace{Ba}_{\text{from } A}, \underbrace{\lambda}_{\text{from } A}, \underbrace{B}_{\text{from } Ba}, \underbrace{Ab}_{\text{from } B}, \underbrace{d}_{\text{from } B}, \underbrace{b}_{\text{from } Ab} \,\}$$

$$\begin{aligned}\text{PreFirst}(B) &= \{\, B, \underbrace{Ab}_{\text{from } B}, \underbrace{d}_{\text{from } B}, \underbrace{b}_{\text{from } Ab}, \underbrace{A}_{\text{from } Ab} \,\} \cup \text{PreFirst}(A)\\ &= \{\, A, Ba, \lambda, B, Ab, d, b \,\}\end{aligned}$$

$$\text{PreFirst}(S) = \{\, S, \underbrace{AAc}_{\text{from } S}, \underbrace{Ac}_{\text{from } AAc}, \underbrace{c}_{\text{from } Ac}, \underbrace{A}_{\text{from } AAc} \,\} \cup \text{PreFirst}(A)$$

Exercise

$$S \to AAc \qquad A \to Ba \mid \lambda \qquad B \to Ab \mid d$$

The erasable variables ($V \Rightarrow^+ \lambda$) are: A .

We determine PreFirst(A), PreFirst(B) and PreFirst(S):

$$\text{PreFirst}(A) = \{\, A, \underbrace{Ba}_{\text{from } A}, \underbrace{\lambda}_{\text{from } A}, \underbrace{B}_{\text{from } Ba}, \underbrace{Ab}_{\text{from } B}, \underbrace{d}_{\text{from } B}, \underbrace{b}_{\text{from } Ab} \,\}$$

$$\text{PreFirst}(B) = \{\, B, \underbrace{Ab}_{\text{from } B}, \underbrace{d}_{\text{from } B}, \underbrace{b}_{\text{from } Ab}, \underbrace{A}_{\text{from } Ab} \,\} \cup \text{PreFirst}(A)$$

$$= \{\, A, Ba, \lambda, B, Ab, d, b \,\}$$

$$\text{PreFirst}(S) = \{\, S, \underbrace{AAc}_{\text{from } S}, \underbrace{Ac}_{\text{from } AAc}, \underbrace{c}_{\text{from } Ac}, \underbrace{A}_{\text{from } AAc} \,\} \cup \text{PreFirst}(A)$$

$$= \{\, S, AAc, Ac, c, A, Ba, \lambda, B, Ab, d, b \,\}$$

Exercise

$$S \to AAc \qquad A \to Ba \mid \lambda \qquad B \to Ab \mid d$$

The erasable variables ($V \Rightarrow^{+} \lambda$) are: A .

We determine PreFirst(A), PreFirst(B) and PreFirst(S):

$$\text{PreFirst}(A) = \{ A, \underbrace{Ba}_{\text{from } A}, \underbrace{\lambda}_{\text{from } A}, \underbrace{B}_{\text{from } Ba}, \underbrace{Ab}_{\text{from } B}, \underbrace{d}_{\text{from } B}, \underbrace{b}_{\text{from } Ab} \}$$

$$\begin{aligned}\text{PreFirst}(B) &= \{ B, \underbrace{Ab}_{\text{from } B}, \underbrace{d}_{\text{from } B}, \underbrace{b}_{\text{from } Ab}, \underbrace{A}_{\text{from } Ab} \} \cup \text{PreFirst}(A) \\ &= \{ A, Ba, \lambda, B, Ab, d, b \}\end{aligned}$$

$$\begin{aligned}\text{PreFirst}(S) &= \{ S, \underbrace{AAc}_{\text{from } S}, \underbrace{Ac}_{\text{from } AAc}, \underbrace{c}_{\text{from } Ac}, \underbrace{A}_{\text{from } AAc} \} \cup \text{PreFirst}(A) \\ &= \{ S, AAc, Ac, c, A, Ba, \lambda, B, Ab, d, b \}\end{aligned}$$

Thus we get

$\text{First}(A) =$ $\qquad$ $\text{First}(B) =$ $\qquad$ $\text{First}(S) =$

Exercise

$$S \to AAc \qquad A \to Ba \mid \lambda \qquad B \to Ab \mid d$$

The erasable variables ($V \Rightarrow^{+} \lambda$) are: A .

We determine PreFirst(A), PreFirst(B) and PreFirst(S):

$$\text{PreFirst}(A) = \{\, A, \underbrace{Ba}_{\text{from } A}, \underbrace{\lambda}_{\text{from } A}, \underbrace{B}_{\text{from } Ba}, \underbrace{Ab}_{\text{from } B}, \underbrace{d}_{\text{from } B}, \underbrace{b}_{\text{from } Ab} \,\}$$

$$\begin{aligned} \text{PreFirst}(B) &= \{\, B, \underbrace{Ab}_{\text{from } B}, \underbrace{d}_{\text{from } B}, \underbrace{b}_{\text{from } Ab}, \underbrace{A}_{\text{from } Ab} \,\} \cup \text{PreFirst}(A) \\ &= \{\, A, Ba, \lambda, B, Ab, d, b \,\} \end{aligned}$$

$$\begin{aligned} \text{PreFirst}(S) &= \{\, S, \underbrace{AAc}_{\text{from } S}, \underbrace{Ac}_{\text{from } AAc}, \underbrace{c}_{\text{from } Ac}, \underbrace{A}_{\text{from } AAc} \,\} \cup \text{PreFirst}(A) \\ &= \{\, S, AAc, Ac, c, A, Ba, \lambda, B, Ab, d, b \,\} \end{aligned}$$

Thus we get

$\text{First}(A) = \{\, b, d, \lambda \,\}$ $\quad$ $\text{First}(B) =$ $\quad$ $\text{First}(S) =$

Exercise

$$S \to AAc \qquad A \to Ba \mid \lambda \qquad B \to Ab \mid d$$

The erasable variables ($V \Rightarrow^+ \lambda$) are: A .

We determine PreFirst(A), PreFirst(B) and PreFirst(S):

$$\text{PreFirst}(A) = \{\, A, \underbrace{Ba}_{\text{from } A}, \underbrace{\lambda}_{\text{from } A}, \underbrace{B}_{\text{from } Ba}, \underbrace{Ab}_{\text{from } B}, \underbrace{d}_{\text{from } B}, \underbrace{b}_{\text{from } Ab} \,\}$$

$$\text{PreFirst}(B) = \{\, B, \underbrace{Ab}_{\text{from } B}, \underbrace{d}_{\text{from } B}, \underbrace{b}_{\text{from } Ab}, \underbrace{A}_{\text{from } Ab} \,\} \cup \text{PreFirst}(A)$$

$$= \{\, A, Ba, \lambda, B, Ab, d, b \,\}$$

$$\text{PreFirst}(S) = \{\, S, \underbrace{AAc}_{\text{from } S}, \underbrace{Ac}_{\text{from } AAc}, \underbrace{c}_{\text{from } Ac}, \underbrace{A}_{\text{from } AAc} \,\} \cup \text{PreFirst}(A)$$

$$= \{\, S, AAc, Ac, c, A, Ba, \lambda, B, Ab, d, b \,\}$$

Thus we get

$$\text{First}(A) = \{\, b, d, \lambda \,\} \quad \text{First}(B) = \{\, b, d \,\} \quad \text{First}(S) =$$

Exercise

$$S \to AAc \qquad A \to Ba \mid \lambda \qquad B \to Ab \mid d$$

The erasable variables ($V \Rightarrow^+ \lambda$) are: A .

We determine PreFirst(A), PreFirst(B) and PreFirst(S):

$$\text{PreFirst}(A) = \{\, A, \underbrace{Ba}_{\text{from } A}, \underbrace{\lambda}_{\text{from } A}, \underbrace{B}_{\text{from } Ba}, \underbrace{Ab}_{\text{from } B}, \underbrace{d}_{\text{from } B}, \underbrace{b}_{\text{from } Ab} \,\}$$

$$\begin{aligned} \text{PreFirst}(B) &= \{\, B, \underbrace{Ab}_{\text{from } B}, \underbrace{d}_{\text{from } B}, \underbrace{b}_{\text{from } Ab}, \underbrace{A}_{\text{from } Ab} \,\} \cup \text{PreFirst}(A) \\ &= \{\, A, Ba, \lambda, B, Ab, d, b \,\} \end{aligned}$$

$$\begin{aligned} \text{PreFirst}(S) &= \{\, S, \underbrace{AAc}_{\text{from } S}, \underbrace{Ac}_{\text{from } AAc}, \underbrace{c}_{\text{from } Ac}, \underbrace{A}_{\text{from } AAc} \,\} \cup \text{PreFirst}(A) \\ &= \{\, S, AAc, Ac, c, A, Ba, \lambda, B, Ab, d, b \,\} \end{aligned}$$

Thus we get

$$\text{First}(A) = \{\, b, d, \lambda \,\} \quad \text{First}(B) = \{\, b, d \,\} \quad \text{First}(S) = \{\, b, c, d \,\}$$

Follow(A)

Follow(A)

The sets First(A) are not yet sufficient for ‘predictive’ parsing, if there are derivations $A \Rightarrow^{+} \lambda$.

Follow(A)

The sets First(A) are not yet sufficient for 'predictive' parsing, if there are derivations $A \Rightarrow^+ \lambda$.

We consider the terminal letters that can follow a variable:

$$\text{Follow}(A) = \{\, a \in T \mid S \Rightarrow^* \ldots Aa \ldots \}$$

Intuition: $a \in$ Follow(A) if A can be followed by a in a derivation.

Follow(A)

The sets First(A) are not yet sufficient for 'predictive' parsing, if there are derivations $A \Rightarrow^+ \lambda$.

We consider the terminal letters that can follow a variable:

$$\text{Follow}(A) = \{\, a \in T \mid S \Rightarrow^* \ldots Aa \ldots \}$$

Intuition: $a \in \text{Follow}(A)$ if A can be followed by a in a derivation.

We use $\$$ as a special '**end of word**' symbol.

Algorithm

Follow(A)

The sets First(A) are not yet sufficient for 'predictive' parsing, if there are derivations $A \Rightarrow^+ \lambda$.

We consider the terminal letters that can follow a variable:

$$\text{Follow}(A) = \{\, a \in T \mid S \Rightarrow^* \ldots Aa \ldots \}$$

Intuition: $a \in$ Follow(A) if A can be followed by a in a derivation.

We use $\$$ as a special '**end of word**' symbol.

Algorithm

- Follow(S) $\supseteq \{\, \$ \,\}$

Follow(A)

The sets First(A) are not yet sufficient for 'predictive' parsing,
if there are derivations $A \Rightarrow^+ \lambda$.

We consider the terminal letters that can follow a variable:

$$\text{Follow}(A) = \{\, a \in T \mid S \Rightarrow^* \ldots Aa \ldots \}$$

Intuition: $a \in$ Follow(A) if A can be followed by a in a derivation.

We use $\$$ as a special '**end of word**' symbol.

Algorithm

- Follow(S) $\supseteq \{\, \$ \,\}$
- Follow(A) $\supseteq$ First(w) $\setminus \{\, \lambda \,\}$ for every rule $B \to vAw$

Follow(A)

The sets First(A) are not yet sufficient for 'predictive' parsing, if there are derivations $A \Rightarrow^{+} \lambda$.

We consider the terminal letters that can follow a variable:

$$\text{Follow}(A) = \{\, a \in T \mid S \Rightarrow^{*} \ldots Aa \ldots \,\}$$

Intuition: $a \in$ Follow(A) if A can be followed by a in a derivation.

We use $\$$ as a special '**end of word**' symbol.

Algorithm

- Follow(S) $\supseteq \{\,\$\,\}$
- Follow(A) $\supseteq$ First(w) $\setminus \{\,\lambda\,\}$ for every rule $B \rightarrow vAw$
- Follow(A) $\supseteq$ Follow(B) for rules $B \rightarrow vAw$ with $\lambda \in$ First(w)

Example

- $\mathrm{Follow}(S) \supseteq \{\$\}$
- $\mathrm{Follow}(A) \supseteq \mathrm{First}(w) \setminus \{\lambda\}$ for every rule $B \to vAw$
- $\mathrm{Follow}(A) \supseteq \mathrm{Follow}(B)$ for rules $B \to vAw$ with $\lambda \in \mathrm{First}(w)$

If $C \to AB$, then:

Example

- $\text{Follow}(S) \supseteq \{\$\}$
- $\text{Follow}(A) \supseteq \text{First}(w) \setminus \{\lambda\}$ for every rule $B \to vAw$
- $\text{Follow}(A) \supseteq \text{Follow}(B)$ for rules $B \to vAw$ with $\lambda \in \text{First}(w)$

If $C \to AB$, then:

- $\text{First}(B) \subseteq \text{Follow}(A)$

  Example: $C \Rightarrow AB \Rightarrow^* Aaw$ if $B \to aw$

Example

- Follow(S) $\supseteq \{\$\}$
- Follow(A) $\supseteq$ First(w) $\setminus \{\lambda\}$ for every rule $B \to vAw$
- Follow(A) $\supseteq$ Follow(B) for rules $B \to vAw$ with $\lambda \in$ First(w)

If $C \to AB$, then:

- First(B) $\subseteq$ Follow(A)

  Example: $C \Rightarrow AB \Rightarrow^* Aaw$ if $B \to aw$

- Follow(C) $\subseteq$ Follow(B)

  Example: $S \Rightarrow Ca \Rightarrow ABa$ if $S \to Ca$

Example

- $\text{Follow}(S) \supseteq \{\$\}$
- $\text{Follow}(A) \supseteq \text{First}(w) \setminus \{\lambda\}$ for every rule $B \to vAw$
- $\text{Follow}(A) \supseteq \text{Follow}(B)$ for rules $B \to vAw$ with $\lambda \in \text{First}(w)$

If $C \to AB$, then:

- $\text{First}(B) \subseteq \text{Follow}(A)$

  Example: $C \Rightarrow AB \Rightarrow^* Aaw$ if $B \to aw$

- $\text{Follow}(C) \subseteq \text{Follow}(B)$

  Example: $S \Rightarrow Ca \Rightarrow ABa$ if $S \to Ca$

- $\text{Follow}(C) \subseteq \text{Follow}(A)$ if $B \Rightarrow^* \lambda$

  Example: $S \Rightarrow Ca \Rightarrow ABa \Rightarrow Aa$ if $S \to Ca$ and $B \to \lambda$

Exercise

- $\mathrm{Follow}(S) \supseteq \{\,\$\,\}$
- $\mathrm{Follow}(A) \supseteq \mathrm{First}(w) \setminus \{\,\lambda\,\}$ for every rule $B \to vAw$
- $\mathrm{Follow}(A) \supseteq \mathrm{Follow}(B)$ for rules $B \to vAw$ with $\lambda \in \mathrm{First}(w)$

$$S \to Dc \qquad\qquad A \to Ba \mid \lambda$$

$$D \to AA \qquad\qquad B \to Ab \mid d$$

We have

$$\mathrm{First}(S) = \{\, b,\ c,\ d \,\} \qquad \mathrm{First}(A) = \{\, \lambda,\ b,\ d \,\}$$

$$\mathrm{First}(D) = \{\, \lambda,\ b,\ d \,\} \qquad \mathrm{First}(B) = \{\, b,\ d \,\}$$

Determine $\mathrm{Follow}(S)$, $\mathrm{Follow}(D)$, $\mathrm{Follow}(A)$, $\mathrm{Follow}(B)$:

Exercise

- $\text{Follow}(S) \supseteq \{\$\}$
- $\text{Follow}(A) \supseteq \text{First}(w) \setminus \{\lambda\}$ for every rule $B \to vAw$
- $\text{Follow}(A) \supseteq \text{Follow}(B)$ for rules $B \to vAw$ with $\lambda \in \text{First}(w)$

$$S \to Dc \qquad A \to Ba \mid \lambda$$
$$D \to AA \qquad B \to Ab \mid d$$

We have

$$\text{First}(S) = \{b, c, d\} \qquad \text{First}(A) = \{\lambda, b, d\}$$
$$\text{First}(D) = \{\lambda, b, d\} \qquad \text{First}(B) = \{b, d\}$$

Determine $\text{Follow}(S)$, $\text{Follow}(D)$, $\text{Follow}(A)$, $\text{Follow}(B)$:

$\text{Follow}(S) \supseteq$

Exercise

- Follow(S) $\supseteq \{\,\$\,\}$
- Follow(A) $\supseteq$ First(w) $\setminus \{\,\lambda\,\}$ for every rule $B \to vAw$
- Follow(A) $\supseteq$ Follow(B) for rules $B \to vAw$ with $\lambda \in$ First(w)

$$S \to Dc \qquad\qquad A \to Ba \mid \lambda$$

$$D \to AA \qquad\qquad B \to Ab \mid d$$

We have

$$\text{First}(S) = \{\,b,\ c,\ d\,\} \qquad \text{First}(A) = \{\,\lambda,\ b,\ d\,\}$$

$$\text{First}(D) = \{\,\lambda,\ b,\ d\,\} \qquad \text{First}(B) = \{\,b,\ d\,\}$$

Determine Follow(S), Follow(D), Follow(A), Follow(B):

$$\text{Follow}(S) \supseteq \{\,\$\,\}$$

Exercise

- $\text{Follow}(S) \supseteq \{\$\}$
- $\text{Follow}(A) \supseteq \text{First}(w) \setminus \{\lambda\}$ for every rule $B \to vAw$
- $\text{Follow}(A) \supseteq \text{Follow}(B)$ for rules $B \to vAw$ with $\lambda \in \text{First}(w)$

$$S \to Dc \qquad\qquad A \to Ba \mid \lambda$$
$$D \to AA \qquad\qquad B \to Ab \mid d$$

We have

$$\text{First}(S) = \{b, c, d\} \qquad \text{First}(A) = \{\lambda, b, d\}$$
$$\text{First}(D) = \{\lambda, b, d\} \qquad \text{First}(B) = \{b, d\}$$

Determine $\text{Follow}(S)$, $\text{Follow}(D)$, $\text{Follow}(A)$, $\text{Follow}(B)$:

$$\text{Follow}(S) \supseteq \{\$\}$$
$$\text{Follow}(D) \supseteq$$

Exercise

- $\text{Follow}(S) \supseteq \{\$\}$
- $\text{Follow}(A) \supseteq \text{First}(w) \setminus \{\lambda\}$ for every rule $B \to vAw$
- $\text{Follow}(A) \supseteq \text{Follow}(B)$ for rules $B \to vAw$ with $\lambda \in \text{First}(w)$

$$S \to Dc \qquad\qquad A \to Ba \mid \lambda$$
$$D \to AA \qquad\qquad B \to Ab \mid d$$

We have

$$\text{First}(S) = \{b, c, d\} \qquad \text{First}(A) = \{\lambda, b, d\}$$
$$\text{First}(D) = \{\lambda, b, d\} \qquad \text{First}(B) = \{b, d\}$$

Determine $\text{Follow}(S)$, $\text{Follow}(D)$, $\text{Follow}(A)$, $\text{Follow}(B)$:

$$\text{Follow}(S) \supseteq \{\$\}$$
$$\text{Follow}(D) \supseteq \{c\}$$

Exercise

- $\text{Follow}(S) \supseteq \{\,\$\,\}$
- $\text{Follow}(A) \supseteq \text{First}(w) \setminus \{\,\lambda\,\}$ for every rule $B \to vAw$
- $\text{Follow}(A) \supseteq \text{Follow}(B)$ for rules $B \to vAw$ with $\lambda \in \text{First}(w)$

$$S \to Dc \qquad\qquad A \to Ba \mid \lambda$$
$$D \to AA \qquad\qquad B \to Ab \mid d$$

We have

$$\text{First}(S) = \{\, b,\ c,\ d \,\} \qquad \text{First}(A) = \{\, \lambda,\ b,\ d \,\}$$
$$\text{First}(D) = \{\, \lambda,\ b,\ d \,\} \qquad \text{First}(B) = \{\, b,\ d \,\}$$

Determine $\text{Follow}(S)$, $\text{Follow}(D)$, $\text{Follow}(A)$, $\text{Follow}(B)$:

$$\text{Follow}(S) \supseteq \{\,\$\,\}$$
$$\text{Follow}(D) \supseteq \{\, c \,\}$$
$$\text{Follow}(A) \supseteq$$

Exercise

- $\text{Follow}(S) \supseteq \{\$\}$
- $\text{Follow}(A) \supseteq \text{First}(w) \setminus \{\lambda\}$ for every rule $B \to vAw$
- $\text{Follow}(A) \supseteq \text{Follow}(B)$ for rules $B \to vAw$ with $\lambda \in \text{First}(w)$

$$S \to Dc \qquad\qquad A \to Ba \mid \lambda$$
$$D \to AA \qquad\qquad B \to Ab \mid d$$

We have

$$\text{First}(S) = \{b, c, d\} \qquad \text{First}(A) = \{\lambda, b, d\}$$
$$\text{First}(D) = \{\lambda, b, d\} \qquad \text{First}(B) = \{b, d\}$$

Determine $\text{Follow}(S)$, $\text{Follow}(D)$, $\text{Follow}(A)$, $\text{Follow}(B)$:

$$\text{Follow}(S) \supseteq \{\$\}$$
$$\text{Follow}(D) \supseteq \{c\}$$
$$\text{Follow}(A) \supseteq (\text{First}(A) \setminus \{\lambda\})$$

Exercise

- $\text{Follow}(S) \supseteq \{\,\$\,\}$
- $\text{Follow}(A) \supseteq \text{First}(w) \setminus \{\,\lambda\,\}$ for every rule $B \to vAw$
- $\text{Follow}(A) \supseteq \text{Follow}(B)$ for rules $B \to vAw$ with $\lambda \in \text{First}(w)$

$$S \to Dc \qquad\qquad A \to Ba \mid \lambda$$
$$D \to AA \qquad\qquad B \to Ab \mid d$$

We have

$$\text{First}(S) = \{\,b,\ c,\ d\,\} \qquad \text{First}(A) = \{\,\lambda,\ b,\ d\,\}$$
$$\text{First}(D) = \{\,\lambda,\ b,\ d\,\} \qquad \text{First}(B) = \{\,b,\ d\,\}$$

Determine $\text{Follow}(S)$, $\text{Follow}(D)$, $\text{Follow}(A)$, $\text{Follow}(B)$:

$$\text{Follow}(S) \supseteq \{\,\$\,\}$$
$$\text{Follow}(D) \supseteq \{\,c\,\}$$
$$\text{Follow}(A) \supseteq (\text{First}(A) \setminus \{\,\lambda\,\}) \cup \{\,b\,\}$$

Exercise

- $\text{Follow}(S) \supseteq \{\,\$\,\}$
- $\text{Follow}(A) \supseteq \text{First}(w) \setminus \{\,\lambda\,\}$ for every rule $B \to vAw$
- $\text{Follow}(A) \supseteq \text{Follow}(B)$ for rules $B \to vAw$ with $\lambda \in \text{First}(w)$

$$S \to Dc \qquad\qquad A \to Ba \mid \lambda$$
$$D \to AA \qquad\qquad B \to Ab \mid d$$

We have

$$\text{First}(S) = \{\, b,\ c,\ d \,\} \qquad \text{First}(A) = \{\, \lambda,\ b,\ d \,\}$$
$$\text{First}(D) = \{\, \lambda,\ b,\ d \,\} \qquad \text{First}(B) = \{\, b,\ d \,\}$$

Determine $\text{Follow}(S)$, $\text{Follow}(D)$, $\text{Follow}(A)$, $\text{Follow}(B)$:

$$\text{Follow}(S) \supseteq \{\,\$\,\}$$
$$\text{Follow}(D) \supseteq \{\, c \,\}$$
$$\text{Follow}(A) \supseteq (\text{First}(A) \setminus \{\,\lambda\,\}) \cup \{\, b \,\} \cup \text{Follow}(D)$$

Exercise

- $\text{Follow}(S) \supseteq \{\,\$\,\}$
- $\text{Follow}(A) \supseteq \text{First}(w) \setminus \{\,\lambda\,\}$ for every rule $B \to vAw$
- $\text{Follow}(A) \supseteq \text{Follow}(B)$ for rules $B \to vAw$ with $\lambda \in \text{First}(w)$

$$S \to Dc \qquad\qquad A \to Ba \mid \lambda$$

$$D \to AA \qquad\qquad B \to Ab \mid d$$

We have

$$\text{First}(S) = \{\,b,\ c,\ d\,\} \qquad \text{First}(A) = \{\,\lambda,\ b,\ d\,\}$$

$$\text{First}(D) = \{\,\lambda,\ b,\ d\,\} \qquad \text{First}(B) = \{\,b,\ d\,\}$$

Determine $\text{Follow}(S)$, $\text{Follow}(D)$, $\text{Follow}(A)$, $\text{Follow}(B)$:

$$\text{Follow}(S) \supseteq \{\,\$\,\}$$

$$\text{Follow}(D) \supseteq \{\,c\,\}$$

$$\text{Follow}(A) \supseteq (\text{First}(A) \setminus \{\,\lambda\,\}) \cup \{\,b\,\} \cup \text{Follow}(D) \supseteq \{\,b, c, d\,\}$$

Exercise

- $\text{Follow}(S) \supseteq \{\$\}$
- $\text{Follow}(A) \supseteq \text{First}(w) \setminus \{\lambda\}$ for every rule $B \to vAw$
- $\text{Follow}(A) \supseteq \text{Follow}(B)$ for rules $B \to vAw$ with $\lambda \in \text{First}(w)$

$$S \to Dc \qquad\qquad A \to Ba \mid \lambda$$
$$D \to AA \qquad\qquad B \to Ab \mid d$$

We have

$$\text{First}(S) = \{b, c, d\} \qquad \text{First}(A) = \{\lambda, b, d\}$$
$$\text{First}(D) = \{\lambda, b, d\} \qquad \text{First}(B) = \{b, d\}$$

Determine $\text{Follow}(S)$, $\text{Follow}(D)$, $\text{Follow}(A)$, $\text{Follow}(B)$:

$$\text{Follow}(S) \supseteq \{\$\}$$
$$\text{Follow}(D) \supseteq \{c\}$$
$$\text{Follow}(A) \supseteq (\text{First}(A) \setminus \{\lambda\}) \cup \{b\} \cup \text{Follow}(D) \supseteq \{b, c, d\}$$
$$\text{Follow}(B) \supseteq$$

Exercise

- $\text{Follow}(S) \supseteq \{\,\$\,\}$
- $\text{Follow}(A) \supseteq \text{First}(w) \setminus \{\,\lambda\,\}$ for every rule $B \to vAw$
- $\text{Follow}(A) \supseteq \text{Follow}(B)$ for rules $B \to vAw$ with $\lambda \in \text{First}(w)$

$$S \to Dc \qquad\qquad A \to Ba \mid \lambda$$
$$D \to AA \qquad\qquad B \to Ab \mid d$$

We have

$$\text{First}(S) = \{\, b,\ c,\ d \,\} \qquad \text{First}(A) = \{\, \lambda,\ b,\ d \,\}$$
$$\text{First}(D) = \{\, \lambda,\ b,\ d \,\} \qquad \text{First}(B) = \{\, b,\ d \,\}$$

Determine $\text{Follow}(S)$, $\text{Follow}(D)$, $\text{Follow}(A)$, $\text{Follow}(B)$:

$$\text{Follow}(S) \supseteq \{\,\$\,\}$$
$$\text{Follow}(D) \supseteq \{\, c \,\}$$
$$\text{Follow}(A) \supseteq (\text{First}(A) \setminus \{\,\lambda\,\}) \cup \{\, b \,\} \cup \text{Follow}(D) \supseteq \{\, b, c, d \,\}$$
$$\text{Follow}(B) \supseteq \{\, a \,\}$$

Parser Tables

Parser Tables

The **parser table** for a context-free grammar is a table with

- columns indexed by terminals $T \cup \{\$\}$,
- rows indexed by variables V,

At place $[a \in T \cup \{\$\}, B \in V]$ it contains rules $B \to u$ for which

- $a \in \text{First}(u)$, or (never the case for $a = \$$)
- $\lambda \in \text{First}(u)$ and $a \in \text{Follow}(B)$.

Parser Tables

The **parser table** for a context-free grammar is a table with

- columns indexed by terminals $T \cup \{\$\}$,
- rows indexed by variables V,

At place $[a \in T \cup \{\$\}, B \in V]$ it contains rules $B \to u$ for which

- $a \in \text{First}(u)$, or (never the case for $a = \$$)
- $\lambda \in \text{First}(u)$ and $a \in \text{Follow}(B)$.

$$S \to aSb \mid \lambda$$

We have

- $\text{First}(aSb) = \{a\}$, $\text{First}(\lambda) = \{\lambda\}$, $\text{First}(S) = \{\lambda, a\}$
- $\text{Follow}(S) = \{b, \$\}$,

Thus the parser table is:

	a	b	$\$$
S			

Parser Tables

The **parser table** for a context-free grammar is a table with

- columns indexed by terminals $T \cup \{\$\}$,
- rows indexed by variables V,

At place $[a \in T \cup \{\$\}, B \in V]$ it contains rules $B \to u$ for which

- $a \in \text{First}(u)$, or (never the case for $a = \$$)
- $\lambda \in \text{First}(u)$ and $a \in \text{Follow}(B)$.

$$S \to aSb \mid \lambda$$

We have

- $\text{First}(aSb) = \{a\}$, $\text{First}(\lambda) = \{\lambda\}$, $\text{First}(S) = \{\lambda, a\}$
- $\text{Follow}(S) = \{b, \$\}$,

Thus the parser table is:

	a	b	$\$$
S	$S \to aSb$		

Parser Tables

The **parser table** for a context-free grammar is a table with

- columns indexed by terminals $T \cup \{\$\}$,
- rows indexed by variables V,

At place $[a \in T \cup \{\$\}, B \in V]$ it contains rules $B \to u$ for which

- $a \in \text{First}(u)$, or (never the case for $a = \$$)
- $\lambda \in \text{First}(u)$ and $a \in \text{Follow}(B)$.

$$S \to aSb \mid \lambda$$

We have

- $\text{First}(aSb) = \{a\}$, $\text{First}(\lambda) = \{\lambda\}$, $\text{First}(S) = \{\lambda, a\}$
- $\text{Follow}(S) = \{b, \$\}$,

Thus the parser table is:

	a	b	$\$$
S	$S \to aSb$	$S \to \lambda$	

Parser Tables

The **parser table** for a context-free grammar is a table with

- columns indexed by terminals $T \cup \{\$\}$,
- rows indexed by variables V,

At place $[a \in T \cup \{\$\}, B \in V]$ it contains rules $B \to u$ for which

- $a \in \text{First}(u)$, or (never the case for $a = \$$)
- $\lambda \in \text{First}(u)$ and $a \in \text{Follow}(B)$.

$$S \to aSb \mid \lambda$$

We have

- $\text{First}(aSb) = \{a\}$, $\text{First}(\lambda) = \{\lambda\}$, $\text{First}(S) = \{\lambda, a\}$
- $\text{Follow}(S) = \{b, \$\}$,

Thus the parser table is:

	a	b	$\$$
S	$S \to aSb$	$S \to \lambda$	$S \to \lambda$

LL(1) Grammars and Parsing

A grammar is LL(1) if its parser table contains in ever cell $[a \in T \cup \{\$\}, B \in V]$ at most one production rule.

LL(1) Grammars and Parsing

A grammar is LL(1) if its parser table contains in ever cell $[a \in T \cup \{\$\}, B \in V]$ at most one production rule.

An LL(1) parser reads from Left to right, performs a Leftmost derivation, and looks always at 1 symbol of the input.

LL(1) Grammars and Parsing

A grammar is LL(1) if its parser table contains in ever cell $[a \in T \cup \{\$\}, B \in V]$ at most one production rule.

An LL(1) parser reads from Left to right, performs a Leftmost derivation, and looks always at 1 symbol of the input.

Given an LL(1)-grammar and parsing table.

LL(1) Grammars and Parsing

A grammar is LL(1) if its parser table contains in ever cell $[a \in T \cup \{\$\}, B \in V]$ at most one production rule.

An LL(1) parser reads from Left to right, performs a Leftmost derivation, and looks always at 1 symbol of the input.

Given an LL(1)-grammar and parsing table.

To parse $a_1 \cdots a_n$, we start with $\langle S\$, a_1 \cdots a_n\$\rangle$.

LL(1) Grammars and Parsing

A grammar is LL(1) if its parser table contains in ever cell $[a \in T \cup \{\$\}, B \in V]$ at most one production rule.

An LL(1) parser reads from Left to right, performs a Leftmost derivation, and looks always at 1 symbol of the input.

Given an LL(1)-grammar and parsing table.

To parse $a_1 \cdots a_n$, we start with $\langle S\$, a_1 \cdots a_n\$\rangle$.

From a state $\langle v, w\rangle$ we can do the following steps:

LL(1) Grammars and Parsing

A grammar is LL(1) if its parser table contains in ever cell $[a \in T \cup \{\$\}, B \in V]$ at most one production rule.

An LL(1) parser reads from Left to right, performs a Leftmost derivation, and looks always at 1 symbol of the input.

Given an LL(1)-grammar and parsing table.

To parse $a_1 \cdots a_n$, we start with $\langle S\$, a_1 \cdots a_n\$\rangle$.

From a state $\langle v, w\rangle$ we can do the following steps:

- $\langle av', aw'\rangle$ becomes $\langle v', w'\rangle$

LL(1) Grammars and Parsing

A grammar is LL(1) if its parser table contains in ever cell $[a \in T \cup \{\$\}, B \in V]$ at most one production rule.

An LL(1) parser reads from Left to right, performs a Leftmost derivation, and looks always at 1 symbol of the input.

Given an LL(1)-grammar and parsing table.

To parse $a_1 \cdots a_n$, we start with $\langle S\$, a_1 \cdots a_n\$\rangle$.

From a state $\langle v, w\rangle$ we can do the following steps:

- $\langle av', aw'\rangle$ becomes $\langle v', w'\rangle$
- $\langle Bv', aw'\rangle$ becomes $\langle uv', aw'\rangle$ if $B \to u$ at position $[a, B]$

LL(1) Grammars and Parsing

A grammar is LL(1) if its parser table contains in ever cell $[a \in T \cup \{\$\}, B \in V]$ at most one production rule.

An LL(1) parser reads from Left to right, performs a Leftmost derivation, and looks always at 1 symbol of the input.

Given an LL(1)-grammar and parsing table.

To parse $a_1 \cdots a_n$, we start with $\langle S\$, a_1 \cdots a_n\$\rangle$.

From a state $\langle v, w\rangle$ we can do the following steps:

- $\langle av', aw'\rangle$ becomes $\langle v', w'\rangle$
- $\langle Bv', aw'\rangle$ becomes $\langle uv', aw'\rangle$ if $B \to u$ at position $[a, B]$
- $\langle Bv', \$\rangle$ becomes $\langle v', \$\rangle$ if $B \to u$ at position $[\$, B]$

LL(1) Grammars and Parsing

A grammar is LL(1) if its parser table contains in ever cell $[a \in T \cup \{\$\}, B \in V]$ at most one production rule.

An LL(1) parser reads from Left to right, performs a Leftmost derivation, and looks always at 1 symbol of the input.

Given an LL(1)-grammar and parsing table.

To parse $a_1 \cdots a_n$, we start with $\langle S\$, a_1 \cdots a_n\$\rangle$.

From a state $\langle v, w\rangle$ we can do the following steps:

- $\langle av', aw'\rangle$ becomes $\langle v', w'\rangle$
- $\langle Bv', aw'\rangle$ becomes $\langle uv', aw'\rangle$ if $B \to u$ at position $[a, B]$
- $\langle Bv', \$\rangle$ becomes $\langle v', \$\rangle$ if $B \to u$ at position $[\$, B]$
- $\langle \$, \$\rangle$ results in **accept**

LL(1) Grammars and Parsing

A grammar is LL(1) if its parser table contains in ever cell $[a \in T \cup \{\$\}, B \in V]$ at most one production rule.

An LL(1) parser reads from Left to right, performs a Leftmost derivation, and looks always at 1 symbol of the input.

Given an LL(1)-grammar and parsing table.

To parse $a_1 \cdots a_n$, we start with $\langle S\$, a_1 \cdots a_n\$\rangle$.

From a state $\langle v, w\rangle$ we can do the following steps:

- $\langle av', aw'\rangle$ becomes $\langle v', w'\rangle$
- $\langle Bv', aw'\rangle$ becomes $\langle uv', aw'\rangle$ if $B \to u$ at position $[a, B]$
- $\langle Bv', \$\rangle$ becomes $\langle v', \$\rangle$ if $B \to u$ at position $[\$, B]$
- $\langle \$, \$\rangle$ results in **accept**
- In all other cases, $\langle v, w\rangle$ results in **reject**!

Example

$$S \rightarrow aSb \mid \lambda$$

The parser table is:

	a	b	$\$$
S	$S \rightarrow aSb$	$S \rightarrow \lambda$	$S \rightarrow \lambda$

$\langle S\$, ab\$\rangle$

Example

$$S \to aSb \mid \lambda$$

The parser table is:

	a	b	$\$$
S	$S \to aSb$	$S \to \lambda$	$S \to \lambda$

$$\langle S\$, ab\$\rangle \to \langle aSb\$, ab\$\rangle$$

Example

$$S \to aSb \mid \lambda$$

The parser table is:

	a	b	$\$$
S	$S \to aSb$	$S \to \lambda$	$S \to \lambda$

$$\langle S\$, ab\$\rangle \to \langle aSb\$, ab\$\rangle \to \langle Sb\$, b\$\rangle$$

Example

$$S \to aSb \mid \lambda$$

The parser table is:

	a	b	$\$$
S	$S \to aSb$	$S \to \lambda$	$S \to \lambda$

$$\langle S\$, ab\$\rangle \to \langle aSb\$, ab\$\rangle \to \langle Sb\$, b\$\rangle \to \langle b\$, b\$\rangle$$

Example

$$S \to aSb \mid \lambda$$

The parser table is:

	a	b	$\$$
S	$S \to aSb$	$S \to \lambda$	$S \to \lambda$

$$\begin{aligned}\langle S\$, ab\$\rangle &\to \langle aSb\$, ab\$\rangle \to \langle Sb\$, b\$\rangle \to \langle b\$, b\$\rangle \\ &\to \langle \$, \$\rangle\end{aligned}$$

Example

$$S \rightarrow aSb \mid \lambda$$

The parser table is:

	a	b	$\$$
S	$S \rightarrow aSb$	$S \rightarrow \lambda$	$S \rightarrow \lambda$

$$\langle S\$, ab\$\rangle \rightarrow \langle aSb\$, ab\$\rangle \rightarrow \langle Sb\$, b\$\rangle \rightarrow \langle b\$, b\$\rangle$$
$$\rightarrow \langle \$, \$\rangle \quad \textbf{accept}$$

Example

$$S \rightarrow aSb \mid \lambda$$

The parser table is:

	a	b	$\$$
S	$S \rightarrow aSb$	$S \rightarrow \lambda$	$S \rightarrow \lambda$

$$\langle S\$, ab\$\rangle \rightarrow \langle aSb\$, ab\$\rangle \rightarrow \langle Sb\$, b\$\rangle \rightarrow \langle b\$, b\$\rangle$$
$$\rightarrow \langle \$, \$\rangle \quad \textbf{accept}$$

$$\langle S\$, abb\$\rangle$$

Example

$$S \rightarrow aSb \mid \lambda$$

The parser table is:

	a	b	$\$$
S	$S \rightarrow aSb$	$S \rightarrow \lambda$	$S \rightarrow \lambda$

$$\langle S\$, ab\$\rangle \rightarrow \langle aSb\$, ab\$\rangle \rightarrow \langle Sb\$, b\$\rangle \rightarrow \langle b\$, b\$\rangle$$
$$\rightarrow \langle \$, \$\rangle \quad \textbf{accept}$$

$$\langle S\$, abb\$\rangle \rightarrow \langle aSb\$, abb\$\rangle$$

Example

$$S \rightarrow aSb \mid \lambda$$

The parser table is:

	a	b	$\$$
S	$S \rightarrow aSb$	$S \rightarrow \lambda$	$S \rightarrow \lambda$

$$\langle S\$, ab\$ \rangle \rightarrow \langle aSb\$, ab\$ \rangle \rightarrow \langle Sb\$, b\$ \rangle \rightarrow \langle b\$, b\$ \rangle$$
$$\rightarrow \langle \$, \$ \rangle \quad \textbf{accept}$$

$$\langle S\$, abb\$ \rangle \rightarrow \langle aSb\$, abb\$ \rangle \rightarrow \langle Sb\$, bb\$ \rangle$$

Example

$$S \rightarrow aSb \mid \lambda$$

The parser table is:

	a	b	$\$$
S	$S \rightarrow aSb$	$S \rightarrow \lambda$	$S \rightarrow \lambda$

$$\langle S\$, ab\$\rangle \rightarrow \langle aSb\$, ab\$\rangle \rightarrow \langle Sb\$, b\$\rangle \rightarrow \langle b\$, b\$\rangle$$
$$\rightarrow \langle \$, \$\rangle \quad \textbf{accept}$$

$$\langle S\$, abb\$\rangle \rightarrow \langle aSb\$, abb\$\rangle \rightarrow \langle Sb\$, bb\$\rangle \rightarrow \langle b\$, bb\$\rangle$$

Example

$$S \to aSb \mid \lambda$$

The parser table is:

	a	b	$\$$
S	$S \to aSb$	$S \to \lambda$	$S \to \lambda$

$$\begin{aligned} \langle S\$, ab\$\rangle &\to \langle aSb\$, ab\$\rangle \to \langle Sb\$, b\$\rangle \to \langle b\$, b\$\rangle \\ &\to \langle \$, \$\rangle \quad \textbf{accept} \end{aligned}$$

$$\begin{aligned} \langle S\$, abb\$\rangle &\to \langle aSb\$, abb\$\rangle \to \langle Sb\$, bb\$\rangle \to \langle b\$, bb\$\rangle \\ &\to \langle \$, b\$\rangle \end{aligned}$$

Example

$$S \to aSb \mid \lambda$$

The parser table is:

	a	b	$\$$
S	$S \to aSb$	$S \to \lambda$	$S \to \lambda$

$$\langle S\$, ab\$\rangle \to \langle aSb\$, ab\$\rangle \to \langle Sb\$, b\$\rangle \to \langle b\$, b\$\rangle$$
$$\to \langle \$, \$\rangle \quad \textbf{accept}$$

$$\langle S\$, abb\$\rangle \to \langle aSb\$, abb\$\rangle \to \langle Sb\$, bb\$\rangle \to \langle b\$, bb\$\rangle$$
$$\to \langle \$, b\$\rangle \quad \textbf{reject}$$

Example

$$S \to aSb \mid \lambda$$

The parser table is:

	a	b	$\$$
S	$S \to aSb$	$S \to \lambda$	$S \to \lambda$

$$\langle S\$, ab\$\rangle \to \langle aSb\$, ab\$\rangle \to \langle Sb\$, b\$\rangle \to \langle b\$, b\$\rangle$$
$$\to \langle \$, \$\rangle \quad \textbf{accept}$$

$$\langle S\$, abb\$\rangle \to \langle aSb\$, abb\$\rangle \to \langle Sb\$, bb\$\rangle \to \langle b\$, bb\$\rangle$$
$$\to \langle \$, b\$\rangle \quad \textbf{reject}$$

$$\langle S\$, aab\$\rangle$$

Example

$$S \rightarrow aSb \mid \lambda$$

The parser table is:

	a	b	$\$$
S	$S \rightarrow aSb$	$S \rightarrow \lambda$	$S \rightarrow \lambda$

$$\langle S\$, ab\$\rangle \rightarrow \langle aSb\$, ab\$\rangle \rightarrow \langle Sb\$, b\$\rangle \rightarrow \langle b\$, b\$\rangle$$
$$\rightarrow \langle \$, \$\rangle \quad \textbf{accept}$$

$$\langle S\$, abb\$\rangle \rightarrow \langle aSb\$, abb\$\rangle \rightarrow \langle Sb\$, bb\$\rangle \rightarrow \langle b\$, bb\$\rangle$$
$$\rightarrow \langle \$, b\$\rangle \quad \textbf{reject}$$

$$\langle S\$, aab\$\rangle \rightarrow \langle aSb\$, aab\$\rangle$$

Example

$$S \to aSb \mid \lambda$$

The parser table is:

	a	b	$\$$
S	$S \to aSb$	$S \to \lambda$	$S \to \lambda$

$$\langle S\$, ab\$\rangle \to \langle aSb\$, ab\$\rangle \to \langle Sb\$, b\$\rangle \to \langle b\$, b\$\rangle$$
$$\to \langle \$, \$\rangle \quad \textbf{accept}$$

$$\langle S\$, abb\$\rangle \to \langle aSb\$, abb\$\rangle \to \langle Sb\$, bb\$\rangle \to \langle b\$, bb\$\rangle$$
$$\to \langle \$, b\$\rangle \quad \textbf{reject}$$

$$\langle S\$, aab\$\rangle \to \langle aSb\$, aab\$\rangle \to \langle Sb\$, ab\$\rangle$$

Example

$$S \to aSb \mid \lambda$$

The parser table is:

	a	b	$\$$
S	$S \to aSb$	$S \to \lambda$	$S \to \lambda$

$$\langle S\$, ab\$\rangle \to \langle aSb\$, ab\$\rangle \to \langle Sb\$, b\$\rangle \to \langle b\$, b\$\rangle$$
$$\to \langle \$, \$\rangle \quad \textbf{accept}$$

$$\langle S\$, abb\$\rangle \to \langle aSb\$, abb\$\rangle \to \langle Sb\$, bb\$\rangle \to \langle b\$, bb\$\rangle$$
$$\to \langle \$, b\$\rangle \quad \textbf{reject}$$

$$\langle S\$, aab\$\rangle \to \langle aSb\$, aab\$\rangle \to \langle Sb\$, ab\$\rangle \to \langle aSbb\$, ab\$\rangle$$

Example

$$S \rightarrow aSb \mid \lambda$$

The parser table is:

	a	b	$\$$
S	$S \rightarrow aSb$	$S \rightarrow \lambda$	$S \rightarrow \lambda$

$$\langle S\$, ab\$\rangle \rightarrow \langle aSb\$, ab\$\rangle \rightarrow \langle Sb\$, b\$\rangle \rightarrow \langle b\$, b\$\rangle$$
$$\rightarrow \langle \$, \$\rangle \quad \textbf{accept}$$

$$\langle S\$, abb\$\rangle \rightarrow \langle aSb\$, abb\$\rangle \rightarrow \langle Sb\$, bb\$\rangle \rightarrow \langle b\$, bb\$\rangle$$
$$\rightarrow \langle \$, b\$\rangle \quad \textbf{reject}$$

$$\langle S\$, aab\$\rangle \rightarrow \langle aSb\$, aab\$\rangle \rightarrow \langle Sb\$, ab\$\rangle \rightarrow \langle aSbb\$, ab\$\rangle$$
$$\rightarrow \langle Sbb\$, b\$\rangle$$

Example

$$S \to aSb \mid \lambda$$

The parser table is:

	a	b	$\$$
S	$S \to aSb$	$S \to \lambda$	$S \to \lambda$

$$\langle S\$, ab\$\rangle \to \langle aSb\$, ab\$\rangle \to \langle Sb\$, b\$\rangle \to \langle b\$, b\$\rangle$$
$$\to \langle \$, \$\rangle \quad \textbf{accept}$$

$$\langle S\$, abb\$\rangle \to \langle aSb\$, abb\$\rangle \to \langle Sb\$, bb\$\rangle \to \langle b\$, bb\$\rangle$$
$$\to \langle \$, b\$\rangle \quad \textbf{reject}$$

$$\langle S\$, aab\$\rangle \to \langle aSb\$, aab\$\rangle \to \langle Sb\$, ab\$\rangle \to \langle aSbb\$, ab\$\rangle$$
$$\to \langle Sbb\$, b\$\rangle \to \langle bb\$, b\$\rangle$$

Example

$$S \to aSb \mid \lambda$$

The parser table is:

	a	b	$\$$
S	$S \to aSb$	$S \to \lambda$	$S \to \lambda$

$$\langle S\$, ab\$\rangle \to \langle aSb\$, ab\$\rangle \to \langle Sb\$, b\$\rangle \to \langle b\$, b\$\rangle$$
$$\to \langle \$, \$\rangle \quad \textbf{accept}$$

$$\langle S\$, abb\$\rangle \to \langle aSb\$, abb\$\rangle \to \langle Sb\$, bb\$\rangle \to \langle b\$, bb\$\rangle$$
$$\to \langle \$, b\$\rangle \quad \textbf{reject}$$

$$\langle S\$, aab\$\rangle \to \langle aSb\$, aab\$\rangle \to \langle Sb\$, ab\$\rangle \to \langle aSbb\$, ab\$\rangle$$
$$\to \langle Sbb\$, b\$\rangle \to \langle bb\$, b\$\rangle \to \langle b\$, \$\rangle$$

Example

$$S \rightarrow aSb \mid \lambda$$

The parser table is:

	a	b	$\$$
S	$S \rightarrow aSb$	$S \rightarrow \lambda$	$S \rightarrow \lambda$

$$\langle S\$, ab\$\rangle \rightarrow \langle aSb\$, ab\$\rangle \rightarrow \langle Sb\$, b\$\rangle \rightarrow \langle b\$, b\$\rangle$$
$$\rightarrow \langle \$, \$\rangle \quad \textbf{accept}$$

$$\langle S\$, abb\$\rangle \rightarrow \langle aSb\$, abb\$\rangle \rightarrow \langle Sb\$, bb\$\rangle \rightarrow \langle b\$, bb\$\rangle$$
$$\rightarrow \langle \$, b\$\rangle \quad \textbf{reject}$$

$$\langle S\$, aab\$\rangle \rightarrow \langle aSb\$, aab\$\rangle \rightarrow \langle Sb\$, ab\$\rangle \rightarrow \langle aSbb\$, ab\$\rangle$$
$$\rightarrow \langle Sbb\$, b\$\rangle \rightarrow \langle bb\$, b\$\rangle \rightarrow \langle b\$, \$\rangle \quad \textbf{reject}$$

Example

$$S \to aSb \mid \lambda$$

The parser table is:

	a	b	$\$$
S	$S \to aSb$	$S \to \lambda$	$S \to \lambda$

$$\langle S\$, ab\$\rangle \to \langle aSb\$, ab\$\rangle \to \langle Sb\$, b\$\rangle \to \langle b\$, b\$\rangle$$
$$\to \langle \$, \$\rangle \quad \textbf{accept}$$

$$\langle S\$, abb\$\rangle \to \langle aSb\$, abb\$\rangle \to \langle Sb\$, bb\$\rangle \to \langle b\$, bb\$\rangle$$
$$\to \langle \$, b\$\rangle \quad \textbf{reject}$$

$$\langle S\$, aab\$\rangle \to \langle aSb\$, aab\$\rangle \to \langle Sb\$, ab\$\rangle \to \langle aSbb\$, ab\$\rangle$$
$$\to \langle Sbb\$, b\$\rangle \to \langle bb\$, b\$\rangle \to \langle b\$, \$\rangle \quad \textbf{reject}$$

JavaCC (Java Compiler Compiler) automatically generates a parser from an LL(1) grammar.

LL(k) Grammars

LL(k) Grammars

The class of LL(1) grammars is often to restrictive in practice.

LL(k) Grammars

The class of LL(1) grammars is often to restrictive in practice.

LL(1) parsers looks at 1 symbol to decide which rule to use.

LL(k) Grammars

The class of LL(1) grammars is often to restrictive in practice.

LL(1) parsers looks at 1 symbol to decide which rule to use.

An LL(k) parser looks k symbols ahead to choose the rule.

LL(k) Grammars

The class of LL(1) grammars is often to restrictive in practice.

LL(1) parsers looks at 1 symbol to decide which rule to use.

An LL(k) parser looks k symbols ahead to choose the rule.

The parser table is constructed with k symbols look-ahead.

LL(k) Grammars

The class of LL(1) grammars is often to restrictive in practice.

LL(1) parsers looks at 1 symbol to decide which rule to use.

An LL(k) parser looks k symbols ahead to choose the rule.

The parser table is constructed with k symbols look-ahead.

A grammar is LL(k) if this table has in every cell $\leq$ 1 rule.

LL(k) Grammars

The class of LL(1) grammars is often to restrictive in practice.

LL(1) parsers looks at 1 symbol to decide which rule to use.

An LL(k) parser looks k symbols ahead to choose the rule.

The parser table is constructed with k symbols look-ahead.

A grammar is LL(k) if this table has in every cell ≤ 1 rule.

LL(k) is strictly contained in LL($k + 1$).

LL(k) Grammars

The class of LL(1) grammars is often to restrictive in practice.

LL(1) parsers looks at 1 symbol to decide which rule to use.

An LL(k) parser looks k symbols ahead to choose the rule.

The parser table is constructed with k symbols look-ahead.

A grammar is LL(k) if this table has in every cell $\leq$ 1 rule.

LL(k) is strictly contained in LL$(k + 1)$.

Disadvantage: size of the parser table grows exponential in k.

Exercises

Can ambiguous grammars be LL(k) for some $k \geq 1$?

Is the following grammar LL(k) for some $k \geq 1$?

$$S \rightarrow aSa \mid \lambda$$

Left Factorisation

Left Factorisation

Left factorisation: rewrite rules $A \rightarrow uv \mid uw$ ($u \neq \lambda$) into

$$A \rightarrow uB \qquad \text{and} \qquad B \rightarrow v \mid w$$

where B is a fresh variable.

Left Factorisation

Left factorisation: rewrite rules $A \to uv \mid uw$ ($u \neq \lambda$) into

$$A \to uB \qquad \text{and} \qquad B \to v \mid w$$

where B is a fresh variable.

The grammar $S \to ab \mid ac$ is **not** LL(1):

	a	b	c	$\$$
S	$S \to ab$ $S \to ac$			

Left Factorisation

Left factorisation: rewrite rules $A \to uv \mid uw$ $(u \neq \lambda)$ into

$$A \to uB \qquad \text{and} \qquad B \to v \mid w$$

where B is a fresh variable.

The grammar $S \to ab \mid ac$ is **not** LL(1):

	a	b	c	$\$$
S	$S \to ab$ $S \to ac$			

After left factorisation we get: $S \to aA \quad A \to b \mid c$

The grammar $S \to aA$, $A \to b \mid c$ **is** LL(1):

	a	b	c	$\$$
S	$S \to aA$			
A		$A \to b$	$A \to c$	