Lecture 1B – Syntactic Structure

Violet Ka I Pun
violet@ifi.uio.no

Programming Language

A large number of strings (typical infinite), is described by

- Syntax
- Semantics

Programming Language

A large number of strings (typical infinite), is described by

- Syntax
  - Specifies how programs in the language are built up
    E.g., context-free grammar, regular grammar, ...
- Semantics

Programming Language

A large number of strings (typical infinite), is described by

- Syntax
  - Specifies how programs in the language are built up
    E.g., context-free grammar, regular grammar, ...
- Semantics
  - Specifies what programs mean
    E.g., Operationals semantics, denotational semantics, ...

Grammar

A *finite* representation of a language, consists of

- **T**erminals
- **N**onterminals
- A **S**tart symbol
- A set of production **R**ules that defines a nonterminal in terms of a sequence of terminals or non-terminals

Grammar

A *finite* representation of a language, consists of

- **T**erminals
- **N**onterminals
- A **S**tart symbol
- A set of production **R**ules that defines a nonterminal in terms of a sequence of terminals or non-terminals

$\langle$ **N**(onterminals), **T**(erminals), **S** $\in$ N, **R**(ules) $\rangle$

Grammar

A *finite* representation of a language, consists of

- **T**erminals
- **N**onterminals
- A **S**tart symbol
- A set of production **R**ules that defines a nonterminal in terms of a sequence of terminals or non-terminals

$$\langle\ \mathbf{N}(\text{onterminals}),\ \mathbf{T}(\text{erminals}),\ \mathbf{S} \in \mathrm{N},\ \mathbf{R}(\text{ules})\ \rangle\ , \quad \text{where } \mathbf{N} \cap \mathbf{T} = \emptyset$$

Grammar

$\langle$ **N**(onterminals), **T**(erminals), **S** $\in$ N, **R**(ules) $\rangle$

Grammar

$\langle$ **N**(onterminals), **T**(erminals), **S** $\in$ N, **R**(ules) $\rangle$

Context-Free Grammar (CFG): $R \subset N \times (N + T)^+$

e.g.: $\langle\{E, Int, Digit\},\ \{0, 1, ..., 9, +, *, (,)\},\ E,\ R\rangle$

Grammar

$$\langle\ \mathbf{N}\text{(onterminals)},\ \mathbf{T}\text{(erminals)},\ \mathbf{S} \in \text{N},\ \mathbf{R}\text{(ules)}\ \rangle$$

Context-Free Grammar (CFG): $R \subset N \times (N + T)^+$

e.g.: $\langle\{E, Int, Digit\},\ \{0, 1, ..., 9, +, *, (,)\},\ E,\ R\rangle$

BNF (Backus-Naur Form) :

$$
\begin{array}{rrcl}
R = & \{\ E & \to & E + E \\
 & E & \to & E * E \\
 & E & \to & (E) \\
 & E & \to & Int \\
 & Int & \to & Digit \\
 & Int & \to & Digit\ Int \\
 & Digit & \to & 0 \\
 & Digit & \to & 1 \\
 & \ldots & \to & \ldots \\
 & Digit & \to & 9 \quad \}
\end{array}
$$

Grammar

$$\langle\ \mathbf{N}\text{(onterminals)},\ \mathbf{T}\text{(erminals)},\ \mathbf{S} \in \text{N},\ \mathbf{R}\text{(ules)}\ \rangle$$

Context-Free Grammar (CFG): $R \subset N \times (N + T)^+$

e.g.: $\langle\{E, Int, Digit\},\ \{0, 1, ..., 9, +, *, (,)\},\ E,\ R\rangle$

BNF (Backus-Naur Form) :

$$
\begin{array}{rrcl}
R = & \{\ E & \rightarrow & E + E \\
& E & \rightarrow & E * E \\
& E & \rightarrow & (E) \\
& E & \rightarrow & Int \\
& Int & \rightarrow & Digit \\
& Int & \rightarrow & Digit\ Int \\
& Digit & \rightarrow & 0 \\
& Digit & \rightarrow & 1 \\
& \ldots & \rightarrow & \ldots \\
& Digit & \rightarrow & 9 \quad \}
\end{array}
$$

EBNF (Extended BNF) :

What is more: { } , [ ] , | , ( )

$$\langle\ \mathbf{N}\text{(onterminals)},\ \mathbf{T}\text{(erminals)},\ \mathbf{S} \in \text{N},\ \mathbf{R}\text{(ules)}\ \rangle$$

Context-Free Grammar (CFG): $R \subset N \times (N + T)^+$

e.g.: $\langle \{E, Int, Digit\},\ \{0, 1, ..., 9, +, *, (,)\},\ E,\ R\rangle$

BNF (Backus-Naur Form) :

$$\begin{aligned} R = \quad \{\ E &\rightarrow E + E \\ E &\rightarrow E * E \\ E &\rightarrow (E) \\ E &\rightarrow Int \\ Int &\rightarrow Digit \\ Int &\rightarrow Digit\ Int \\ Digit &\rightarrow 0 \\ Digit &\rightarrow 1 \\ \ldots &\rightarrow \ldots \\ Digit &\rightarrow 9 \quad \} \end{aligned}$$

EBNF (Extended BNF) :

What is more: { } , [ ] , | , ( )

$$Digit \rightarrow 0 \mid 1 \mid \ldots \mid 9$$

$$\langle\ \mathbf{N}(\text{onterminals}),\ \mathbf{T}(\text{erminals}),\ \mathbf{S} \in \text{N},\ \mathbf{R}(\text{ules})\ \rangle$$

Context-Free Grammar (CFG): $R \subset N \times (N + T)^{+}$

e.g.: $\langle\{E, Int, Digit\},\ \{0, 1, ..., 9, +, *, (,)\},\ E,\ R\rangle$

BNF (Backus-Naur Form) :

$$
\begin{array}{rrcl}
R = & \{\ E & \rightarrow & E + E \\
 & E & \rightarrow & E * E \\
 & E & \rightarrow & (E) \\
 & E & \rightarrow & Int \\
 & Int & \rightarrow & Digit \\
 & Int & \rightarrow & Digit\ Int \\
 & Digit & \rightarrow & 0 \\
 & Digit & \rightarrow & 1 \\
 & \dots & \rightarrow & \dots \\
 & Digit & \rightarrow & 9 \quad \}
\end{array}
$$

EBNF (Extended BNF) :

$$E \rightarrow E\ (+ \mid *)\ E \mid (E) \mid Int$$

What is more: { } , [ ] , | , ( )

$$Digit \rightarrow 0 \mid 1 \mid \dots \mid 9$$

Grammar

$$\langle \ \mathbf{N}\text{(onterminals)},\ \mathbf{T}\text{(erminals)},\ \mathbf{S} \in \text{N},\ \mathbf{R}\text{(ules)}\ \rangle$$

Context-Free Grammar (CFG): $R \subset N \times (N + T)^{+}$

e.g.: $\langle \{E, Int, Digit\},\ \{0, 1, ..., 9, +, *, (,)\},\ E,\ R\rangle$

BNF (Backus-Naur Form) :

$$
\begin{array}{rrcl}
R = & \{ \ E & \rightarrow & E + E \\
 & E & \rightarrow & E * E \\
 & E & \rightarrow & (E) \\
 & E & \rightarrow & Int \\
 & Int & \rightarrow & Digit \\
 & Int & \rightarrow & Digit\ Int \\
 & Digit & \rightarrow & 0 \\
 & Digit & \rightarrow & 1 \\
 & \dots & \rightarrow & \dots \\
 & Digit & \rightarrow & 9 \quad \}
\end{array}
$$

EBNF (Extended BNF) :

$$E \rightarrow E\ (+ \mid *)\ E \mid (E) \mid Int$$

What is more: { } , [ ] , | , ( )

$$Int \rightarrow Digit\{Digit\}$$

$$Digit \rightarrow 0 \mid 1 \mid \dots \mid 9$$

$$\langle\ \mathbf{N}(\text{onterminals}),\ \mathbf{T}(\text{erminals}),\ \mathbf{S} \in \mathrm{N},\ \mathbf{R}(\text{ules})\ \rangle$$

Context-Free Grammar (CFG): $R \subset N \times (N + T)^+$

e.g.: $\langle\{E, Int, Digit\},\ \{0, 1, ..., 9, +, *, (,)\},\ E,\ R\rangle$

BNF (Backus-Naur Form) :

$$
\begin{array}{rrcl}
R = & \{\ E & \to & E + E \\
 & E & \to & E * E \\
 & E & \to & (E) \\
 & E & \to & Int \\
 & Int & \to & Digit \\
 & Int & \to & Digit\ Int \\
 & Digit & \to & 0 \\
 & Digit & \to & 1 \\
 & \ldots & \to & \ldots \\
 & Digit & \to & 9 \quad \}
\end{array}
$$

EBNF (Extended BNF) :

$$E \to E\ (+ \mid *)\ E \mid (E) \mid Int$$

What is more: { } , [ ] , | , ( )

$$Int \to Digit\{Digit\}$$

$$Digit \to 0 \mid 1 \mid \ldots \mid 9$$

Real number?

Grammar

$$\langle\ \mathbf{N}\text{(onterminals)},\ \mathbf{T}\text{(erminals)},\ \mathbf{S} \in \mathrm{N},\ \mathbf{R}\text{(ules)}\ \rangle$$

Context-Free Grammar (CFG): $R \subset N \times (N + T)^+$

e.g.: $\langle \{E, Int, Digit\},\ \{0, 1, ..., 9, +, *, (,)\},\ E,\ R \rangle$

BNF (Backus-Naur Form) :

$$
\begin{array}{rrcl}
R = & \{\ E & \rightarrow & E + E \\
 & E & \rightarrow & E * E \\
 & E & \rightarrow & (E) \\
 & E & \rightarrow & Int \\
 & Int & \rightarrow & Digit \\
 & Int & \rightarrow & Digit\ Int \\
 & Digit & \rightarrow & 0 \\
 & Digit & \rightarrow & 1 \\
 & \dots & \rightarrow & \dots \\
 & Digit & \rightarrow & 9 \quad \}
\end{array}
$$

EBNF (Extended BNF) :

$$E \rightarrow E\ (+ \mid *)\ E \mid (E) \mid Int$$

What is more: { } , [ ] , | , ( )

$$Int \rightarrow Digit\{Digit\}$$

$$Digit \rightarrow 0 \mid 1 \mid \dots \mid 9$$

Real number?

$$\dots\dots\dots Int.Int \mid .Int$$

$$\langle\ \mathbf{N}(\text{onterminals}),\ \mathbf{T}(\text{erminals}),\ \mathbf{S} \in \text{N},\ \mathbf{R}(\text{ules})\ \rangle$$

Context-Free Grammar (CFG): $R \subset N \times (N + T)^{+}$

e.g.: $\langle\{E, Int, Digit\},\ \{0, 1, ..., 9, +, *, (,)\},\ E,\ R\rangle$

BNF (Backus-Naur Form) :

$$\begin{aligned} R = \quad \{\ E &\rightarrow E + E \\ E &\rightarrow E * E \\ E &\rightarrow (E) \\ E &\rightarrow Int \\ Int &\rightarrow Digit \\ Int &\rightarrow Digit\ Int \\ Digit &\rightarrow 0 \\ Digit &\rightarrow 1 \\ \ldots &\rightarrow \ldots \\ Digit &\rightarrow 9 \quad \} \end{aligned}$$

EBNF (Extended BNF) :

$$E \rightarrow E\ (+ \mid *)\ E \mid (E) \mid Int$$

What is more: { } , [ ] , | , ( )

$$Int \rightarrow Digit\{Digit\}$$

$$Digit \rightarrow 0 \mid 1 \mid \ldots \mid 9$$

Real number?

$\ldots\ldots\ldots Int.Int \mid .Int \ldots\ldots\ldots:\quad [Int].Int$

Alternative Representation of Grammars

BNF :

$$R = \begin{array}{ll} \{ & E \rightarrow E + E \\ & E \rightarrow E * E \\ & E \rightarrow (E) \\ & E \rightarrow Int \end{array}$$

Alternative Representation of Grammars

BNF :

$$R = \left\{ \begin{array}{lcl} E & \rightarrow & E + E \\ E & \rightarrow & E * E \\ E & \rightarrow & (E) \\ E & \rightarrow & Int \end{array} \right.$$

EBNF :

$$E \rightarrow E \; (+ \mid *) \; E \mid (E) \mid Int$$

Alternative Representation of Grammars

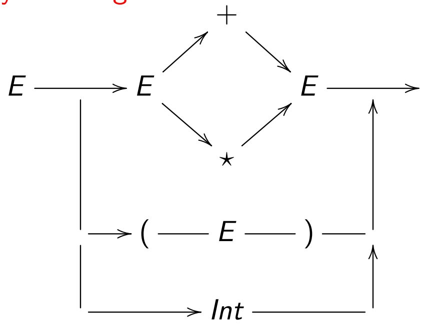
BNF :

$$R = \left\{ \begin{array}{lcl} E & \rightarrow & E + E \\ E & \rightarrow & E * E \\ E & \rightarrow & (E) \\ E & \rightarrow & Int \end{array} \right.$$

EBNF :

$$E \rightarrow E\ (+ \mid *)\ E \mid (E) \mid Int$$

Syntax diagrams



Alternative Representation of Grammars

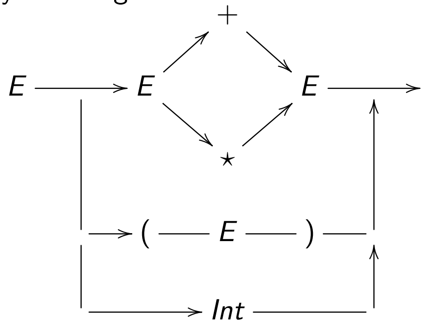
BNF :

$$
\begin{aligned}
R = \quad \{\; E &\rightarrow E + E \\
E &\rightarrow E * E \\
E &\rightarrow (E) \\
E &\rightarrow \mathit{Int} \\
\color{red}{\mathit{Int}} &\rightarrow \color{red}{\mathit{Digit}} \\
\color{red}{\mathit{Int}} &\rightarrow \color{red}{\mathit{Digit}\ \mathit{Int}}
\end{aligned}
$$

EBNF :

$$E \rightarrow E\ (+ \mid *)\ E \mid (E) \mid \mathit{Int}$$

Syntax diagrams



Alternative Representation of Grammars

BNF :

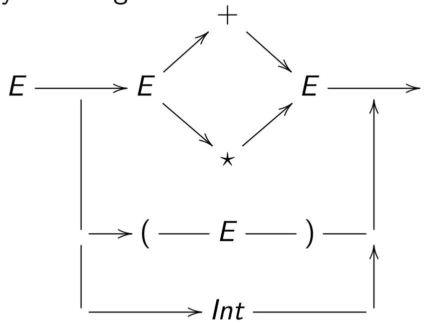
$$
\begin{aligned}
R = \quad \{\ E &\rightarrow E + E \\
E &\rightarrow E * E \\
E &\rightarrow (E) \\
E &\rightarrow Int \\
{\color{red} Int} &\rightarrow {\color{red} Digit} \\
{\color{red} Int} &\rightarrow {\color{red} Digit\ Int}
\end{aligned}
$$

EBNF :

$$E \rightarrow E\ (+ \mid *)\ E \mid (E) \mid Int$$

$${\color{red} Int \rightarrow Digit\{Digit\}}$$

Syntax diagrams



Alternative Representation of Grammars

BNF :

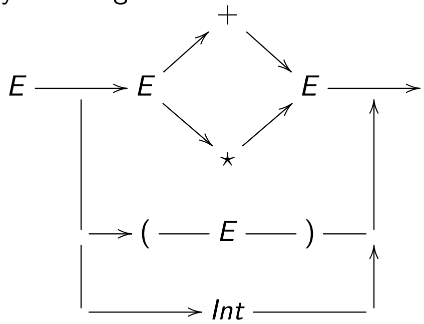
$$
\begin{array}{rrcl}
R = & \{\ E & \rightarrow & E + E \\
 & E & \rightarrow & E * E \\
 & E & \rightarrow & (E) \\
 & E & \rightarrow & \mathit{Int} \\
 & \mathit{Int} & \rightarrow & \mathit{Digit} \\
 & \mathit{Int} & \rightarrow & \mathit{Digit}\ \mathit{Int}
\end{array}
$$

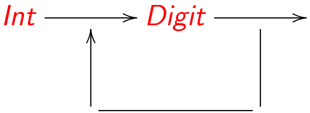
EBNF :

$$E \rightarrow E\ (+ \mid *)\ E \mid (E) \mid \mathit{Int}$$

$$\mathit{Int} \rightarrow \mathit{Digit}\{\mathit{Digit}\}$$

Syntax diagrams





Expression Notations

Expression Notations

- Prefix
- Postfix
- Infix

Expression Notations

- Prefix
- Postfix
- Infix

Programming languages use a mix of them

Pre- and Postfix Notations (Polish and Reverse Polish)

Prefix

- **op** E_1 E_2

Postfix

- E_1 E_2 **op**

Pre- and Postfix Notations (Polish and Reverse Polish)

Prefix

- **op** E_1 E_2

  $* + 2\ 3\ 4$

Postfix

- E_1 E_2 **op**

Pre- and Postfix Notations (Polish and Reverse Polish)

Prefix

- **op** E_1 E_2

  $* \ + \ 2 \ 3 \ 4 = * \ 5 \ 4 \ = 20$

Postfix

- E_1 E_2 **op**

Pre- and Postfix Notations (Polish and Reverse Polish)

Prefix

- **op** E_1 E_2

  $* \ + \ 2\ 3\ 4 = *\ 5\ 4 \ = 20$

  $*\ 2 \ +3\ 4$

Postfix

- E_1 E_2 **op**

Pre- and Postfix Notations (Polish and Reverse Polish)

Prefix

- **op** E_1 E_2

  $* \; + \; 2 \; 3 \; 4 = * \; 5 \; 4 \; = 20$

  $* \; 2 \; + 3 \; 4 = * \; 2 \; 7 \; = 14$

Postfix

- E_1 E_2 **op**

Pre- and Postfix Notations (Polish and Reverse Polish)

Prefix

- **op** E_1 E_2

  $* \ + \ 2\ 3\ 4 = *\ 5\ 4 \ = 20$

  $*\ 2 \ +3\ 4 = *\ 2\ 7 \ = 14$

Postfix

- E_1 E_2 **op**

  $2\ 3\ 4\ + \ *$

Pre- and Postfix Notations (Polish and Reverse Polish)

Prefix

- **op** E_1 E_2

  $* \ + \ 2\ 3\ 4 = *\ 5\ 4 \ = 20$

  $*\ 2 \ +3\ 4 = *\ 2\ 7 \ = 14$

Postfix

- E_1 E_2 **op**

  $2\ 3\ 4 \ + \ * \ = 2\ 7 * \ = 14$

Pre- and Postfix Notations (Polish and Reverse Polish)

Prefix

- **op** E_1 E_2

  $* \; + \; 2 \; 3 \; 4 = * \; 5 \; 4 \; = 20$

  $* \; 2 \; + 3 \; 4 = * \; 2 \; 7 \; = 14$

Postfix

- E_1 E_2 **op**

  $2 \; 3 \; 4 \; + \; * \; = 2 \; 7 \; * \; = 14$

  $2 \; 3 \; + \; 4 \; *$

Pre- and Postfix Notations (Polish and Reverse Polish)

Prefix

- **op** E_1 E_2

  $* \ + \ 2 \ 3 \ 4 = * \ 5 \ 4 \ = 20$

  $* \ 2 \ + 3 \ 4 = * \ 2 \ 7 \ = 14$

Postfix

- E_1 E_2 **op**

  $2 \ 3 \ 4 \ + \ * \ = 2 \ 7 \ * \ = 14$

  $2 \ 3 \ + \ 4 \ * \ = 5 \ 4 \ * \ = 20$

Pre- and Postfix Notations (Polish and Reverse Polish)

Prefix

- **op** E_1 E_2

  $* \ + \ 2 \ 3 \ 4 = * \ 5 \ 4 \ = 20$

  $* \ 2 \ + 3 \ 4 = * \ 2 \ 7 \ = 14$

Postfix

- E_1 E_2 **op**

  $2 \ 3 \ 4 \ + \ * \ = 2 \ 7 \ * \ = 14$

  $2 \ 3 \ + \ 4 \ * \ = 5 \ 4 \ * \ = 20$

Parenthesis-free

Infix Notation

- E_1 **op** E_2
- Ambigious to decode, e.g.,

Infix Notation

- E_1 **op** E_2
- Ambigious to decode, e.g.,
  $4 - 2 - 1$ can be

Infix Notation

- E_1 **op** E_2
- Ambigious to decode, e.g.,

  $4 - 2 - 1$ can be

  - $2 - 1 = 1$, or
  - $4 - 1 = 3$ $\leftarrow$ incorrect

Infix Notation

- E_1 **op** E_2
- Ambigious to decode, e.g.,

  $4 - 2 - 1$ can be

  - $2 - 1 = 1$, or
  - $4 - 1 = 3$ $\leftarrow$ incorrect

  $2 + 3 * 4$ can be

Infix Notation

- E_1 **op** E_2
- Ambigious to decode, e.g.,

  $4 - 2 - 1$ can be

  - $2 - 1 = 1$, or
  - $4 - 1 = 3$ $\leftarrow$ incorrect

  $2 + 3 * 4$ can be

  - $5 * 4 = 20$, or $\leftarrow$ incorrect
  - $2 + 12 = 14$

Infix Notation

- E_1 **op** E_2
- Ambigious to decode, e.g.,

  $4 - 2 - 1$ can be

  - $2 - 1 = 1$, or
  - $4 - 1 = 3$ $\leftarrow$ incorrect

  $2 + 3 * 4$ can be

  - $5 * 4 = 20$, or $\leftarrow$ incorrect
  - $2 + 12 = 14$

- Need rules for associativity and precedence

An operator can be either left or right associative

Left associative

Right associative

Associativity

An operator can be either left or right associative

Left associative

E.g., -,+,*,/ : $1 - 2 - 3 = (1 - 2) - 3 \neq 1 - (2 - 3)$

$$E \to E - Int \mid Int$$

Right associative

Associativity

An operator can be either left or right associative

Left associative

E.g., -,+,*,/ : $1 - 2 - 3 = (1 - 2) - 3 \neq 1 - (2 - 3)$

$$E \rightarrow E - Int \mid Int$$

Right associative

E.g., expression $2^{3^2} = 2^{(3^2)} \neq (2^3)^2$

$$E \rightarrow Int^E \mid Int$$

Associativity

An operator can be either left or right associative

Left associative

E.g., -,+,*,/ : $1 - 2 - 3 = (1 - 2) - 3 \neq 1 - (2 - 3)$

$E \rightarrow E - Int \mid Int$

Right associative

E.g., expression $2^{3^2} = 2^{(3^2)} \neq (2^3)^2$

$E \rightarrow Int^E \mid Int$

Both left and right associative??

Associativity

An operator can be either left or right associative

Left associative

E.g., -,+,*,/ : $1 - 2 - 3 = (1 - 2) - 3 \neq 1 - (2 - 3)$

$E \rightarrow E - Int \mid Int$

Right associative

E.g., expression $2^{3^2} = 2^{(3^2)} \neq (2^3)^2$

$E \rightarrow Int^E \mid Int$

Both left and right associative??

What about ...

$2 + 3 * 4$ can be

- $5 * 4 = 20$, or
- $2 + 12 = 14$

An operator can be either left or right associative

Left associative

E.g., -,+,*,/ : $1 - 2 - 3 = (1 - 2) - 3 \neq 1 - (2 - 3)$

$E \rightarrow E - Int \mid Int$

Right associative

E.g., expression $2^{3^2} = 2^{(3^2)} \neq (2^3)^2$

$E \rightarrow Int^E \mid Int$

Both left and right associative??

What about . . .

$2 + 3 * 4$ can be

- $5 * 4 = 20$, or $\leftarrow$ incorrect
- $2 + 12 = 14$

Precedence

Each precedence level requires a new non-terminal symbol, plus an extra symbol for the lowest elements:

$$
\begin{aligned}
E &\rightarrow E + E \\
E &\rightarrow \ldots \\
E &\rightarrow E * E \\
E &\rightarrow Int \mid (E)
\end{aligned}
$$

Precedence

Each precedence level requires a new non-terminal symbol,
plus an extra symbol for the lowest elements:

$$
\begin{array}{lll}
E \rightarrow E + E & \qquad E \rightarrow G + E \mid G & \text{weakest} \\
E \rightarrow \ldots & \qquad .. \rightarrow \ldots & \\
E \rightarrow E * E & \qquad G \rightarrow L * G \mid L & \text{strongest} \\
E \rightarrow Int \mid (E) & \qquad L \rightarrow Int \mid (E) &
\end{array}
$$

Precedence

Each precedence level requires a new non-terminal symbol,
plus an extra symbol for the lowest elements:

$E \to E + E$	$E \to G + E \mid G$	weakest
$E \to ...$	$.. \to ...$	
$E \to E * E$	$G \to L * G \mid L$	strongest
$E \to Int \mid (E)$	$L \to Int \mid (E)$	

Derive: 2 * 3 + 4 and 2 + 3 * 4

Precedence

Each precedence level requires a new non-terminal symbol,
plus an extra symbol for the lowest elements:

$$\begin{array}{llll} E \to E + E & \qquad E \to G + E \mid G & \text{weakest} \\ E \to \ldots & \qquad .. \to \ldots & \\ E \to E * E & \qquad G \to L * G \mid L & \text{strongest} \\ E \to Int \mid (E) & \qquad L \to Int \mid (E) & \end{array}$$

Derive: 2 * 3 + 4 and 2 + 3 * 4

Associativity

Precedence

Each precedence level requires a new non-terminal symbol,
plus an extra symbol for the lowest elements:

$E \rightarrow E + E$	$E \rightarrow G + E \mid G$	weakest
$E \rightarrow ...$	$.. \rightarrow ...$	
$E \rightarrow E * E$	$G \rightarrow L * G \mid L$	strongest
$E \rightarrow Int \mid (E)$	$L \rightarrow Int \mid (E)$	

Derive: 2 * 3 + 4 and 2 + 3 * 4

Associativity

left: $E \rightarrow E - Int$

Precedence

Each precedence level requires a new non-terminal symbol,
plus an extra symbol for the lowest elements:

$E \rightarrow E + E$	$E \rightarrow G + E \mid G$	weakest
$E \rightarrow ...$	$.. \rightarrow ...$	
$E \rightarrow E * E$	$G \rightarrow L * G \mid L$	strongest
$E \rightarrow Int \mid (E)$	$L \rightarrow Int \mid (E)$	

Derive: 2 * 3 + 4 and 2 + 3 * 4

Associativity

left: $E \rightarrow E - Int$ vs. right: $E \rightarrow Int^{E} \mid Int$

Each precedence level requires a new non-terminal symbol,
plus an extra symbol for the lowest elements:

$E \rightarrow E + E$	$E \rightarrow G + E \mid G$	weakest
$E \rightarrow ...$	$.. \rightarrow ...$	
$E \rightarrow E * E$	$G \rightarrow L * G \mid L$	strongest
$E \rightarrow Int \mid (E)$	$L \rightarrow Int \mid (E)$	

Derive: 2 * 3 + 4 and 2 + 3 * 4

Associativity

left: $E \rightarrow E - Int$ vs. right: $E \rightarrow Int^E \mid Int$

On each level, all the symbols must associate equally

$E \rightarrow P * E \mid E + P$, with $P, E \rightarrow ...Int$, gives ambiguity:

Precedence

Each precedence level requires a new non-terminal symbol,
plus an extra symbol for the lowest elements:

$E \to E + E$	$E \to G + E \mid G$	weakest
$E \to ...$	$.. \to ...$	
$E \to E * E$	$G \to L * G \mid L$	strongest
$E \to Int \mid (E)$	$L \to Int \mid (E)$	

Derive: 2 * 3 + 4 and 2 + 3 * 4

Associativity

left: $E \to E - Int$ vs. right: $E \to Int^E \mid Int$

On each level, all the symbols must associate equally

$E \to P * E \mid E + P$, with $P, E \to ...Int$, gives ambiguity:

2 * 3 + 4 can be (2 * 3) +4 or 2 * (3 + 4)

Mixfix Notation

Programming languages use a mix of prefix, infix and postfix notations

Mixfix Notation

Programming languages use a mix of prefix, infix and postfix notations

```
f(x:int) {
  if x == 1
    then return x++;
    else return ++x;
}
```

Regular Language

$B \rightarrow b \mid B\ B$

Regular Language

$B \to b \mid B\ B$ vs. $B \to b \mid b\ B$

Regular Language

B $\rightarrow$ b | B B   vs.   B $\rightarrow$ b | b B

Regular language = can be defined with a **regular grammar**,
i.e., $\langle N, T, S \in N, R \rangle$, where each rule in R is of one of these forms:

Regular Language

B $\rightarrow$ b | B B   vs.   B $\rightarrow$ b | b B

Regular language = can be defined with a **regular grammar**,
i.e., $\langle N, T, S \in N, R \rangle$, where each rule in R is of one of these forms:
$N \rightarrow a$ for some $a \in T$

Regular Language

B $\rightarrow$ b | B B   vs.   B $\rightarrow$ b | b B

Regular language = can be defined with a **regular grammar**,
i.e., $\langle N, T, S \in N, R \rangle$, where each rule in R is of one of these forms:

$N \rightarrow a$ for some $a \in T$

$N \rightarrow aN$ (or, equivalently, $N \rightarrow Na$)

Regular Language

B $\rightarrow$ b | B B   vs.   B $\rightarrow$ b | b B

Regular language = can be defined with a **regular grammar**,
i.e., $\langle N, T, S \in N, R \rangle$, where each rule in R is of one of these forms:

$N \rightarrow a$ for some $a \in T$

$N \rightarrow aN$ (or, equivalently, $N \rightarrow Na$)

$N \rightarrow \varepsilon$

- E.g..:

Regular Language

B $\rightarrow$ b | B B   vs.   B $\rightarrow$ b | b B

Regular language = can be defined with a **regular grammar**,
i.e., $\langle N, T, S \in N, R \rangle$, where each rule in R is of one of these forms:

$N \rightarrow a$	for some $a \in T$
$N \rightarrow aN$	(or, equivalently, $N \rightarrow Na$)
$N \rightarrow \varepsilon$	

- E.g..:

$A \rightarrow aA \mid \varepsilon$

B $\rightarrow$ b | B B   vs.   B $\rightarrow$ b | b B

Regular language = can be defined with a **regular grammar**,
i.e., $\langle N, T, S \in N, R \rangle$, where each rule in R is of one of these forms:

$N \rightarrow a$ for some $a \in T$

$N \rightarrow aN$ (or, equivalently, $N \rightarrow Na$)

$N \rightarrow \varepsilon$

- E.g..:

$A \rightarrow aA \mid \varepsilon$

$B \rightarrow bB \mid b$

Regular Language

B $\rightarrow$ b | B B vs. B $\rightarrow$ b | b B

Regular language = can be defined with a **regular grammar**,
i.e., $\langle N, T, S \in N, R \rangle$, where each rule in R is of one of these forms:

$N \rightarrow a$ for some $a \in T$

$N \rightarrow aN$ (or, equivalently, $N \rightarrow Na$)

$N \rightarrow \varepsilon$

- E.g..:

$A \rightarrow aA \mid \varepsilon$

$B \rightarrow bB \mid b$

$T \rightarrow aA \mid bA \mid A$

Regular Language

B $\rightarrow$ b | B B   vs.   B $\rightarrow$ b | b B

Regular language = can be defined with a **regular grammar**,
i.e., $\langle N, T, S \in N, R \rangle$, where each rule in R is of one of these forms:

$N \rightarrow a$ for some $a \in T$
$N \rightarrow aN$ (or, equivalently, $N \rightarrow Na$)
$N \rightarrow \varepsilon$

- E.g..:

$A \rightarrow aA \mid \varepsilon$

$B \rightarrow bB \mid b$

$T \rightarrow aA \mid bA \mid A$

$R \rightarrow aR \mid bR \mid a \mid b$

Regular Language

B $\rightarrow$ b | B B   vs.   B $\rightarrow$ b | b B

Regular language = can be defined with a **regular grammar**,
i.e., $\langle N, T, S \in N, R \rangle$, where each rule in R is of one of these forms:

$N \rightarrow a$ for some $a \in T$

$N \rightarrow aN$ (or, equivalently, $N \rightarrow Na$)

$N \rightarrow \varepsilon$

- E.g..:   common notation:

$A \rightarrow aA \mid \varepsilon$

$B \rightarrow bB \mid b$

$T \rightarrow aA \mid bA \mid A$

$R \rightarrow aR \mid bR \mid a \mid b$

B → b | B B   vs.   B → b | b B

Regular language = can be defined with a **regular grammar**,
i.e., $\langle N, T, S \in N, R \rangle$, where each rule in R is of one of these forms:

$N \rightarrow a$ for some $a \in T$

$N \rightarrow aN$ (or, equivalently, $N \rightarrow Na$)

$N \rightarrow \varepsilon$

- E.g..:   common notation:

$A \rightarrow aA \mid \varepsilon$ . . . . a*

$B \rightarrow bB \mid b$

$T \rightarrow aA \mid bA \mid A$

$R \rightarrow aR \mid bR \mid a \mid b$

$\mathrm{B} \to \mathrm{b} \mid \mathrm{B}\ \mathrm{B}$ vs. $\mathrm{B} \to \mathrm{b} \mid \mathrm{b}\ \mathrm{B}$

Regular language = can be defined with a **regular grammar**,
i.e., $\langle N, T, S \in N, R\rangle$, where each rule in R is of one of these forms:

$N \to a$ for some $a \in T$
$N \to aN$ (or, equivalently, $N \to Na$)
$N \to \varepsilon$

- E.g..: common notation:

$A \to aA \mid \varepsilon$ . . . . a*

$B \to bB \mid b$ . . . . b$^+$

$T \to aA \mid bA \mid A$

$R \to aR \mid bR \mid a \mid b$

Regular Language

B $\rightarrow$ b | B B   vs.   B $\rightarrow$ b | b B

Regular language = can be defined with a **regular grammar**,
i.e., $\langle N, T, S \in N, R\rangle$, where each rule in R is of one of these forms:

$N \rightarrow a$ for some $a \in T$

$N \rightarrow aN$ (or, equivalently, $N \rightarrow Na$)

$N \rightarrow \varepsilon$

- E.g..: common notation:

$A \rightarrow aA \mid \varepsilon$ . . . . a*

$B \rightarrow bB \mid b$ . . . . b$^+$

$T \rightarrow aA \mid bA \mid A$ . . . . (a + b) a*

$R \rightarrow aR \mid bR \mid a \mid b$

Regular Language

B $\rightarrow$ b | B B   vs.   B $\rightarrow$ b | b B

Regular language = can be defined with a **regular grammar**,
i.e., $\langle N, T, S \in N, R\rangle$, where each rule in R is of one of these forms:

$N \rightarrow a$	for some $a \in T$
$N \rightarrow aN$	(or, equivalently, $N \rightarrow Na$)
$N \rightarrow \varepsilon$	

- E.g..: common notation:

$A \rightarrow aA \mid \varepsilon$ .... a*

$B \rightarrow bB \mid b$ .... b$^+$

$T \rightarrow aA \mid bA \mid A$ .... (a + b) a*

$R \rightarrow aR \mid bR \mid a \mid b$ .... (a + b)$^+$

B $\rightarrow$ b | B B   vs.   B $\rightarrow$ b | b B

Regular language = can be defined with a **regular grammar**,
i.e., $\langle N, T, S \in N, R \rangle$, where each rule in R is of one of these forms:

$N \rightarrow a$ for some $a \in T$

$N \rightarrow aN$ (or, equivalently, $N \rightarrow Na$)

$N \rightarrow \varepsilon$

- E.g..:   common notation:

  $A \rightarrow aA \mid \varepsilon$ . . . . a*

  $B \rightarrow bB \mid b$ . . . . b$^+$

  $T \rightarrow aA \mid bA \mid A$ . . . . (a + b) a*

  $R \rightarrow aR \mid bR \mid a \mid b$ . . . . (a + b)$^+$

- Regular grammar is always finite

Regular Language

B $\to$ b | B B   vs.   B $\to$ b | b B

Regular language = can be defined with a **regular grammar**,
i.e., $\langle N, T, S \in N, R \rangle$, where each rule in R is of one of these forms:

$N \to a$ for some $a \in T$
$N \to aN$ (or, equivalently, $N \to Na$)
$N \to \varepsilon$

- E.g..:   common notation:

  $A \to aA \mid \varepsilon$ .... a*

  $B \to bB \mid b$ .... b$^+$

  $T \to aA \mid bA \mid A$ .... (a + b) a*

  $R \to aR \mid bR \mid a \mid b$ .... (a + b)$^+$

- Regular grammar is always finite

- Finite-state machines (FSM/FSA)

Ensure Unambiguity

z := 8 – 2 * 3 ;

Ensure Unambiguity

```
z := 8 – 2 * 3 ;
x := z + 7 ;
```

Ensure Unambiguity

z := 8 – 2 * 3 ;
x := z + 7 ;
x := y := z := 5

Ensure Unambiguity

z := 8 – 2 * 3 ;
x := z + 7 ;
x := y := z := 5

$$\text{Pr} \to \text{Var} := \text{Pr} \mid \text{Pr} ; \text{Pr} \mid \text{E}$$

Ensure Unambiguity

z := 8 – 2 * 3 ;
x := z + 7 ;
x := y := z := 5

$$Pr \rightarrow Var := Pr \mid Pr \; ; \; Pr \mid E$$

$$E \rightarrow E \; (+ \mid - \mid * \mid / \;) \; E$$

Ensure Unambiguity

z := 8 – 2 * 3 ;
x := z + 7 ;
x := y := z := 5

$$\text{Pr} \rightarrow \text{Var} := \text{Pr} \mid \text{Pr} \; ; \; \text{Pr} \mid \text{E}$$

$$\text{E} \rightarrow \text{E} \; (+ \mid - \mid * \mid / \;) \; \text{E} \mid \text{Var} \mid \text{int} \mid (\text{E})$$

Ensure Unambiguity

z := 8 – 2 * 3 ;
x := z + 7 ;
x := y := z := 5

Ambiguities!

$$\text{Pr} \rightarrow \text{Var} := \text{Pr} \mid \text{Pr} \; ; \; \text{Pr} \mid \text{E}$$

$$\text{E} \rightarrow \text{E} \; (+ \mid - \mid * \mid / \;) \; \text{E} \mid \text{Var} \mid \text{int} \mid (\text{E})$$

Ensure Unambiguity

z := 8 – 2 * 3 ;
x := z + 7 ;
x := y := z := 5

Ambiguities!

$$\text{Pr} \rightarrow \text{Var} := \text{Pr} \mid \text{Pr} \; ; \; \text{Pr} \mid \text{E}$$

$$\text{E} \rightarrow \text{E} \; (+ \mid - \mid * \mid / \;) \; \text{E} \mid \text{Var} \mid \text{int} \mid (\text{E})$$

x := y := 5 : x := (y := 5) or (x := y) := 5 ?

Ensure Unambiguity

z := 8 – 2 * 3 ;
x := z + 7 ;
x := y := z := 5

$$\text{Pr} \to \text{Var} := \text{Pr} \mid \text{Pr} \,;\, \text{Pr} \mid \text{E}$$

$$\text{E} \to \text{E} \; (+ \mid - \mid * \mid / \,) \; \text{E} \mid \text{Var} \mid \text{int} \mid (\text{E})$$

Ambiguities!

x := y := 5 :   x := (y := 5)   or   (x := y) := 5   ?

x := 5 ; z := 3 :

Ensure Unambiguity

z := 8 – 2 * 3 ;
x := z + 7 ;
x := y := z := 5

$$\mathrm{Pr} \to \mathrm{Var} := \mathrm{Pr} \mid \mathrm{Pr} \; ; \; \mathrm{Pr} \mid \mathrm{E}$$

$$\mathrm{E} \to \mathrm{E} \; (+ \mid - \mid * \mid / \;) \; \mathrm{E} \mid \mathrm{Var} \mid \mathrm{int} \mid (\mathrm{E})$$

Ambiguities!

x := y := 5 :	x := (y := 5)	or	(x := y) := 5	?
x := 5 ; z := 3 :	(x := 5) ; (z := 3)	or	x := (5 ; z := 3)	?

Ensure Unambiguity

z := 8 – 2 * 3 ;
x := z + 7 ;
x := y := z := 5

$$Pr \rightarrow Var := Pr \mid Pr\ ;\ Pr \mid E$$

$$E \rightarrow E\ (\ + \mid - \mid * \mid /\ )\ E \mid Var \mid int \mid (E)$$

Ambiguities!

x := y := 5 :	x := (y := 5)	or	(x := y) := 5	?
x := 5 ; z := 3 :	(x := 5) ; (z := 3)	or	x := (5 ; z := 3)	?
8 – 2 * 3 :	(8 – 2) * 3	or	8 – (2 * 3)	?

Ensure Unambiguity

z := 8 – 2 * 3 ;
x := z + 7 ;
x := y := z := 5

$$\mathrm{Pr} \rightarrow \mathrm{Var} := \mathrm{Pr} \mid \mathrm{Pr} \; ; \; \mathrm{Pr} \mid \mathrm{E}$$

$$\mathrm{E} \rightarrow \mathrm{E} \; (+ \mid - \mid * \mid /) \; \mathrm{E} \mid \mathrm{Var} \mid \mathrm{int} \mid (\mathrm{E})$$

Ambiguities!

x := y := 5 :	x := (y := 5)	or	(x := y) := 5	?
x := 5 ; z := 3 :	(x := 5) ; (z := 3)	or	x := (5 ; z := 3)	?
8 – 2 * 3 :	(8 – 2) * 3	or	8 – (2 * 3)	?
P1 ; P2 ; P3 :	(P1 ; P2) ; P3	or	P1 ; (P2 ; P3)	?

Ensure Unambiguity

z := 8 – 2 * 3 ;
x := z + 7 ;
x := y := z := 5

$$Pr \rightarrow Var := Pr \mid Pr \,;\, Pr \mid E$$

$$E \rightarrow E \; (+ \mid - \mid * \mid /) \; E \mid Var \mid int \mid (E)$$

Ambiguities!

x := y := 5 :	x := (y := 5)	or	(x := y) := 5	?
x := 5 ; z := 3 :	(x := 5) ; (z := 3)	or	x := (5 ; z := 3)	?
8 – 2 * 3 :	(8 – 2) * 3	or	8 – (2 * 3)	?
P1 ; P2 ; P3 :	(P1 ; P2) ; P3	or	P1 ; (P2 ; P3)	?
8 * 2 * 2 :	(8 * 2) * 2	or	8 * (2 * 2)	?

Ensure Unambiguity

z := 8 – 2 * 3 ;
x := z + 7 ;
x := y := z := 5

$$Pr \rightarrow Var := Pr \mid Pr \; ; \; Pr \mid E$$

$$E \rightarrow E \; (+ \mid - \mid * \mid / \;) \; E \mid Var \mid int \mid (E)$$

Ambiguities!

x := y := 5 :	x := (y := 5)	or	(x := y) := 5	?
x := 5 ; z := 3 :	(x := 5) ; (z := 3)	or	x := (5 ; z := 3)	?
8 – 2 * 3 :	(8 – 2) * 3	or	8 – (2 * 3)	?
P1 ; P2 ; P3 :	(P1 ; P2) ; P3	or	P1 ; (P2 ; P3)	?
8 * 2 * 2 :	(8 * 2) * 2	or	8 * (2 * 2)	?
8 – 2 – 2 :	(8 – 2) – 2	or	8 – (2 – 2)	?

Ensure Unambiguity

z := 8 – 2 * 3 ;
x := z + 7 ;
x := y := z := 5

$$Pr \rightarrow Var := Pr \mid Pr \; ; \; Pr \mid E$$

$$E \rightarrow E \; (+ \mid - \mid * \mid /) \; E \mid Var \mid int \mid (E)$$

Ambiguities!

Precedence:

x := y := 5 :	x := (y := 5)	or	(x := y) := 5	?
x := 5 ; z := 3 :	(x := 5) ; (z := 3)	or	x := (5 ; z := 3)	?
8 – 2 * 3 :	(8 – 2) * 3	or	8 – (2 * 3)	?
P1 ; P2 ; P3 :	(P1 ; P2) ; P3	or	P1 ; (P2 ; P3)	?
8 * 2 * 2 :	(8 * 2) * 2	or	8 * (2 * 2)	?
8 – 2 – 2 :	(8 – 2) – 2	or	8 – (2 – 2)	?

Ensure Unambiguity

z := 8 − 2 * 3 ;
x := z + 7 ;
x := y := z := 5

$$\text{Pr} \rightarrow \text{Var} := \text{Pr} \mid \text{Pr} \,;\, \text{Pr} \mid \text{E}$$

$$\text{E} \rightarrow \text{E} \;(\, + \mid - \mid * \mid / \,)\; \text{E} \mid \text{Var} \mid \text{int} \mid (\text{E})$$

Ambiguities!

Precedence:

x := y := 5 :	x := (y := 5)	or	(x := y) := 5	?
x := 5 ; z := 3 :	(x := 5) ; (z := 3)	or	x := (5 ; z := 3)	?
8 − 2 * 3 :	(8 − 2) * 3	or	8 − (2 * 3)	?

Associativity:

P1 ; P2 ; P3 :	(P1 ; P2) ; P3	or	P1 ; (P2 ; P3)	?
8 * 2 * 2 :	(8 * 2) * 2	or	8 * (2 * 2)	?
8 − 2 − 2 :	(8 − 2) − 2	or	8 − (2 − 2)	?

Ensure Unambiguity

$$\mathsf{Pr} \rightarrow \mathsf{V} := \mathsf{Pr} \mid \mathsf{Pr} \; ; \; \mathsf{Pr} \mid \mathsf{E}$$
$$\mathsf{E} \rightarrow \mathsf{E} \; (+ \mid - \mid * \mid / \,) \; \mathsf{E} \mid \mathsf{V} \mid \mathsf{int} \mid (\mathsf{E})$$

Ensure Unambiguity

$$\text{Pr} \rightarrow \text{V} := \text{Pr} \mid \text{Pr} \text{ ; } \text{Pr} \mid \text{E}$$
$$\text{E} \rightarrow \text{E} \text{ (} + \mid - \mid * \mid / \text{) } \text{E} \mid \text{V} \mid \text{int} \mid \text{(E)}$$

	prec.
_;_	0
:=	1
+,–	2
*, /	3

Ensure Unambiguity

$$\mathsf{Pr} \rightarrow \mathsf{V} := \mathsf{Pr} \mid \mathsf{Pr} \;;\; \mathsf{Pr} \mid \mathsf{E}$$
$$\mathsf{E} \rightarrow \mathsf{E}\ (\ + \mid - \mid * \mid /\ )\ \mathsf{E} \mid \mathsf{V} \mid \mathsf{int} \mid (\mathsf{E})$$

	prec.	assos.
_;_	0	
:=	1	
+,–	2	
*, /	3	

Ensure Unambiguity

$$\mathsf{Pr} \rightarrow \mathsf{V} := \mathsf{Pr} \mid \mathsf{Pr} ; \mathsf{Pr} \mid \mathsf{E}$$
$$\mathsf{E} \rightarrow \mathsf{E} \; (+ \mid - \mid * \mid /) \; \mathsf{E} \mid \mathsf{V} \mid \mathsf{int} \mid (\mathsf{E})$$

	prec.	assos.
_;_	0	right
:=	1	right
+,–	2	
*, /	3	

Ensure Unambiguity

$$\mathsf{Pr} \rightarrow \mathsf{V} := \mathsf{Pr} \mid \mathsf{Pr}\ ;\ \mathsf{Pr} \mid \mathsf{E}$$
$$\mathsf{E} \rightarrow \mathsf{E}\ (\ + \mid - \mid * \mid /\ )\ \mathsf{E} \mid \mathsf{V} \mid \mathsf{int} \mid (\mathsf{E})$$

	prec.	assos.
_;_	0	right
:=	1	right
+,–	2	left
*, /	3	left

Ensure Unambiguity

$$\text{Pr} \rightarrow \text{V} := \text{Pr} \mid \text{Pr} \; ; \; \text{Pr} \mid \text{E}$$
$$\text{E} \rightarrow \text{E} \; (+ \mid - \mid * \mid / \;) \; \text{E} \mid \text{V} \mid \text{int} \mid (\text{E})$$

	prec.	assos.
_;_	0	right
:=	1	right
+,–	2	left
*, /	3	left

$$\text{Pr} \quad \rightarrow \quad \text{St} \mid \text{St} \; ; \; \text{Pr}$$

Ensure Unambiguity

$$\mathsf{Pr} \rightarrow \mathsf{V} := \mathsf{Pr} \mid \mathsf{Pr} \; ; \; \mathsf{Pr} \mid \mathsf{E}$$
$$\mathsf{E} \rightarrow \mathsf{E} \; (+ \mid - \mid * \mid / \;) \; \mathsf{E} \mid \mathsf{V} \mid \mathsf{int} \mid (\mathsf{E})$$

	prec.	assos.			
_;_	0	right	Pr	$\rightarrow$	St | St ; Pr
:=	1	right	St	$\rightarrow$	V := St | E
+,–	2	left			
*, /	3	left			

Ensure Unambiguity

$$\text{Pr} \rightarrow \text{V} := \text{Pr} \mid \text{Pr} \,;\, \text{Pr} \mid \text{E}$$
$$\text{E} \rightarrow \text{E} \;(\, + \mid - \mid * \mid / \,)\; \text{E} \mid \text{V} \mid \text{int} \mid (\text{E})$$

	prec.	assos.			
_;_	0	right	Pr	$\rightarrow$	St | St ; Pr
:=	1	right	St	$\rightarrow$	V := St | E
+,–	2	left	E	$\rightarrow$	E + M |E – M | M
*, /	3	left			

Ensure Unambiguity

$$\mathsf{Pr} \to \mathsf{V} := \mathsf{Pr} \mid \mathsf{Pr} \ ; \ \mathsf{Pr} \mid \mathsf{E}$$
$$\mathsf{E} \to \mathsf{E} \ (+ \mid - \mid * \mid / \ ) \ \mathsf{E} \mid \mathsf{V} \mid \mathsf{int} \mid (\mathsf{E})$$

	prec.	assos.			
_;_	0	right	Pr	$\to$	St | St ; Pr
:=	1	right	St	$\to$	V := St | E
+,–	2	left	E	$\to$	E + M |E – M | M
*, /	3	left	M	$\to$	M * F | M / F | F
			F	$\to$	int | V | (E)

Ensure Unambiguity

$$\mathsf{Pr} \rightarrow \mathsf{V} := \mathsf{Pr} \mid \mathsf{Pr}\ ;\ \mathsf{Pr} \mid \mathsf{E}$$
$$\mathsf{E} \rightarrow \mathsf{E}\ (\ + \mid - \mid * \mid /\ )\ \mathsf{E} \mid \mathsf{V} \mid \mathsf{int} \mid (\mathsf{E})$$

	prec.	assos.			
_;_	0	right	Pr	$\rightarrow$	St | St ; Pr
:=	1	right	St	$\rightarrow$	V := St | E
+,–	2	left	E	$\rightarrow$	E + M |E – M | M
*, /	3	left	M	$\rightarrow$	M * F | M / F | F
			F	$\rightarrow$	int | V | (E)

Ensure Unambiguity

$$\mathsf{Pr} \rightarrow \mathsf{V} := \mathsf{Pr} \mid \mathsf{Pr}\ ;\ \mathsf{Pr} \mid \mathsf{E}$$
$$\mathsf{E} \rightarrow \mathsf{E}\ (\ + \mid - \mid * \mid /\ )\ \mathsf{E} \mid \mathsf{V} \mid \mathsf{int} \mid (\mathsf{E})$$

	prec.	assos.			
_;_	0	right	Pr	$\rightarrow$	St | St ; Pr
:=	1	right	St	$\rightarrow$	V := St | V := E
+,–	2	left	E	$\rightarrow$	E + M |E – M | M
*, /	3	left	M	$\rightarrow$	M * F | M / F | F
			F	$\rightarrow$	int | V | (E)

Parse Tree (Concrete Syntax Tree)

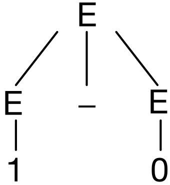
- A representation of grammars in a tree-like form
- Pictorially shows how the start symbol of a grammar derives a string in the language
- Provide syntactic information

Parse Tree (Concrete Syntax Tree)

- A representation of grammars in a tree-like form
- Pictorially shows how the start symbol of a grammar derives a string in the language
- Provide syntactic information

E ::= E − E | 0 | 1

Expression: 1 − 0

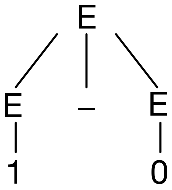


Parse Tree (Concrete Syntax Tree)

- A representation of grammars in a tree-like form
- Pictorially shows how the start symbol of a grammar derives a string in the language
- Provide syntactic information

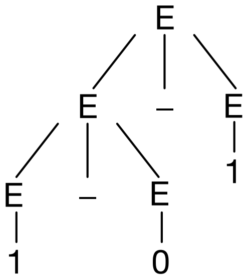
E ::= E − E | 0 | 1

Expression: 1 − 0



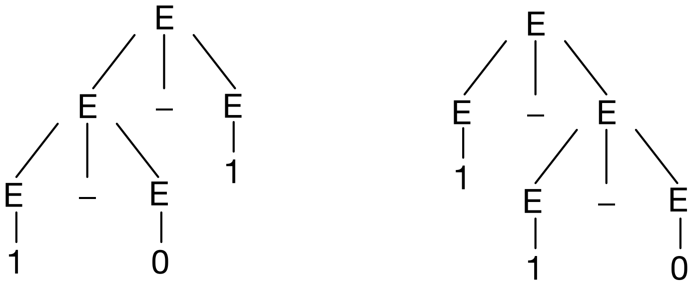
What about for 1 − 0 − 1

What about for $1 - 0 - 1$



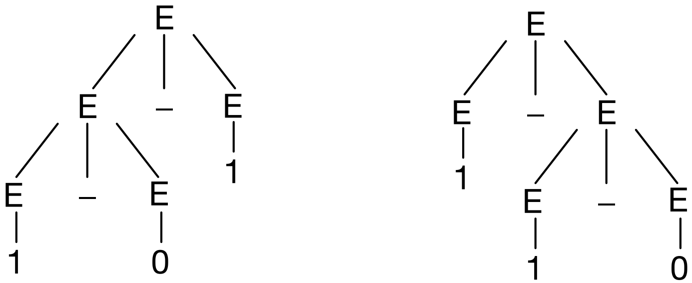
Syntactical Ambiguity

What about for $1 - 0 - 1$



Syntactical Ambiguity

What about for $1 - 0 - 1$



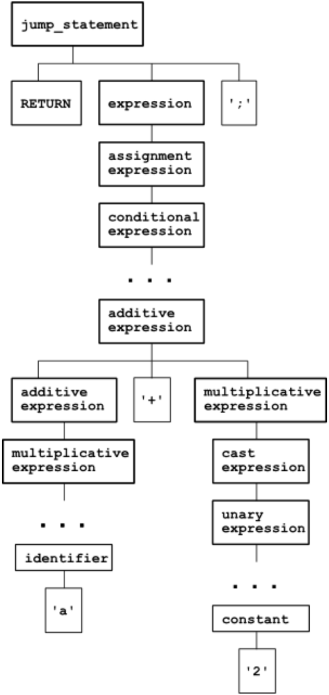
Two or more parse trees $\Rightarrow$ ambigious

Parse trees can be very complicated:

E.g., the C statement, **return** a + 2;

Parse trees can be very complicated:

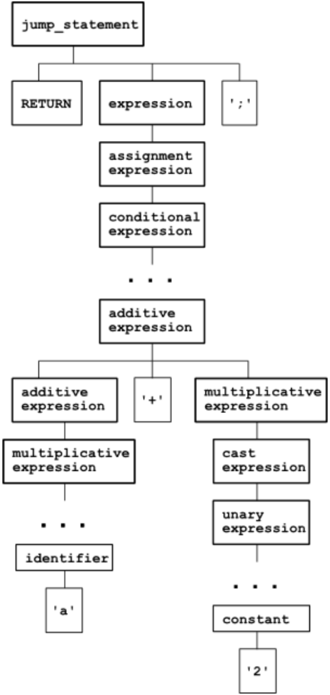
E.g., the C statement, **return** a + 2;



Parse trees can be very complicated:

E.g., the C statement, **return** a + 2;

- Hard to analyse, and processing

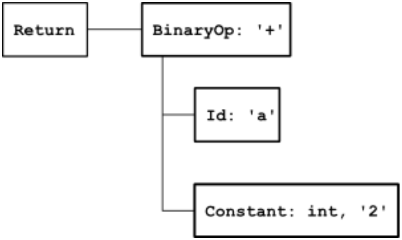


Abstract Syntax Tree (AST)

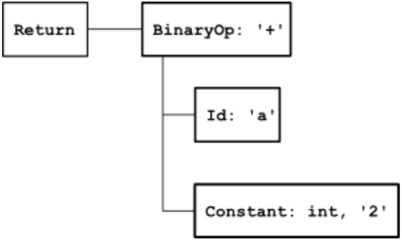
- A simplified syntactic representation of the source code
- Different from parse tree:
  - Does not show the whole syntactic mess
  - Pictorises the parsed string in a structured way

Abstract Syntax Tree (AST)

- A simplified syntactic representation of the source code
- Different from parse tree:
  - Does not show the whole syntactic mess
  - Pictorises the parsed string in a structured way



- A simplified syntactic representation of the source code
- Different from parse tree:
  - Does not show the whole syntactic mess
  - Pictorises the parsed string in a structured way



CST to AST

$$\mathsf{Pr} \rightarrow \mathsf{V} := \mathsf{Pr} \mid \mathsf{Pr} \ ; \ \mathsf{Pr} \mid \mathsf{E}$$
$$\mathsf{E} \rightarrow \mathsf{E} \ (+ \mid - \mid * \mid / \ ) \ \mathsf{E} \mid \mathsf{V} \mid \mathsf{int} \mid (\mathsf{E})$$

$$\mathsf{Pr} \to \mathsf{V} := \mathsf{Pr} \mid \mathsf{Pr}\ ;\ \mathsf{Pr} \mid \mathsf{E}$$
$$\mathsf{E} \to \mathsf{E}\ (\ + \mid - \mid * \mid /\ )\ \mathsf{E} \mid \mathsf{V} \mid \mathsf{int} \mid (\mathsf{E})$$

	prec.	assos.			
_;_	0	right	Pr	$\to$	St | St ; Pr
:=	1	right	St	$\to$	V := St | V := E
+,–	2	left	E	$\to$	E + M | E – M | M
*, /	3	left	M	$\to$	M * F | M / F | F
			F	$\to$	int | V | (E)

Remove left recursion (generally)

$$\mathsf{Pr} \to \mathsf{V} := \mathsf{Pr} \mid \mathsf{Pr} \ ; \ \mathsf{Pr} \mid \mathsf{E}$$
$$\mathsf{E} \to \mathsf{E} \ (+ \mid - \mid * \mid / \ ) \ \mathsf{E} \mid \mathsf{V} \mid \mathsf{int} \mid (\mathsf{E})$$

	prec.	assos.			
_;_	0	right	Pr	$\to$	St | St ; Pr
:=	1	right	St	$\to$	V := St | V := E
+,–	2	left	E	$\to$	E + M | E – M | M
*, /	3	left	M	$\to$	M * F | M / F | F
			F	$\to$	int | V | (E)

Remove left recursion (generally)

E $\to$ **E Rs** | **Trm**

$$\mathrm{Pr} \to \mathrm{V} := \mathrm{Pr} \mid \mathrm{Pr}\ ;\ \mathrm{Pr} \mid \mathrm{E}$$
$$\mathrm{E} \to \mathrm{E}\ (\ +\ \mid\ -\ \mid\ *\ \mid\ /\ )\ \mathrm{E} \mid \mathrm{V} \mid \mathrm{int} \mid (\mathrm{E})$$

	prec.	assos.			
_;_	0	right	Pr	$\to$	St | St ; Pr
:=	1	right	St	$\to$	V := St | V := E
+,–	2	left	E	$\to$	E + M | E – M | M
*, /	3	left	M	$\to$	M * F | M / F | F
			F	$\to$	int | V | (E)

Remove left recursion (generally)

E $\to$ **E Rs** | **Trm** becomes **E** $\to$ **Trm New** and **New** $\to$ **Rs New** | ε

$$\mathsf{Pr} \to \mathsf{V} := \mathsf{Pr} \mid \mathsf{Pr}\ ;\ \mathsf{Pr} \mid \mathsf{E}$$
$$\mathsf{E} \to \mathsf{E}\ (\ + \mid - \mid * \mid /\ )\ \mathsf{E} \mid \mathsf{V} \mid \mathsf{int} \mid (\mathsf{E})$$

	prec.	assos.			
_;_	0	right	Pr	$\to$	St | St ; Pr
:=	1	right	St	$\to$	V := St | V := E
+,–	2	left	E	$\to$	E + M | E – M | M
*, /	3	left	M	$\to$	M * F | M / F | F
			F	$\to$	int | V | (E)

Remove left recursion (generally)

E $\to$ **E Rs** | **Trm** becomes **E** $\to$ **Trm New** and **New** $\to$ **Rs New** | ε

e.g., E $\to$ M New

$$\mathsf{Pr} \rightarrow \mathsf{V} := \mathsf{Pr} \mid \mathsf{Pr} \; ; \; \mathsf{Pr} \mid \mathsf{E}$$
$$\mathsf{E} \rightarrow \mathsf{E} \; (+ \mid - \mid * \mid / \;) \; \mathsf{E} \mid \mathsf{V} \mid \mathsf{int} \mid (\mathsf{E})$$

	prec.	assos.			
_;_	0	right	Pr	$\rightarrow$	St | St ; Pr
:=	1	right	St	$\rightarrow$	V := St | V := E
+,–	2	left	E	$\rightarrow$	E + M | E – M | M
*, /	3	left	M	$\rightarrow$	M * F | M / F | F
			F	$\rightarrow$	int | V | (E)

Remove left recursion (generally)

E $\rightarrow$ **E Rs** | **Trm** becomes **E** $\rightarrow$ **Trm New** and **New** $\rightarrow$ **Rs New** | ε

e.g., E $\rightarrow$ M New and New $\rightarrow$ + M New | – M New | ε

$$\mathsf{Pr} \to \mathsf{V} := \mathsf{Pr} \mid \mathsf{Pr} \; ; \; \mathsf{Pr} \mid \mathsf{E}$$
$$\mathsf{E} \to \mathsf{E} \; (+ \mid - \mid * \mid / \;) \; \mathsf{E} \mid \mathsf{V} \mid \mathsf{int} \mid (\mathsf{E})$$

	prec.	assos.			
_;_	0	right	Pr	$\to$	St | St ; Pr
:=	1	right	St	$\to$	V := St | V := E
+,–	2	left	E	$\to$	E + M | E – M | M
*, /	3	left	M	$\to$	M * F | M / F | F
			F	$\to$	int | V | (E)

Remove left recursion (generally)

E $\to$ **E Rs** | **Trm** becomes **E** $\to$ **Trm New** and **New** $\to$ **Rs New** | ε

e.g., E $\to$ M New and New $\to$ + M New | – M New | ε

change associativity to right!

$$\mathsf{Pr} \to \mathsf{V} := \mathsf{Pr} \mid \mathsf{Pr} \ ; \ \mathsf{Pr} \mid \mathsf{E}$$
$$\mathsf{E} \to \mathsf{E} \ (+ \mid - \mid * \mid / \ ) \ \mathsf{E} \mid \mathsf{V} \mid \mathsf{int} \mid (\mathsf{E})$$

	prec.	assos.			
_;_	0	right	Pr	$\to$	St | St ; Pr
:=	1	right	St	$\to$	V := St | V := E
+,–	2	left	E	$\to$	E + M | E – M | M
*, /	3	left	M	$\to$	M * F | M / F | F
			F	$\to$	int | V | (E)

Remove left recursion (generally)

E $\to$ **E Rs** | **Trm** becomes **E** $\to$ **Trm New** and **New** $\to$ **Rs New** | ε

e.g., E $\to$ M New and New $\to$ + M New | – M New | ε

change associativity to right! – manipulation of left-recursion requires something more than 'recursive descent parsing'