Intermediate Code Generation

Sudakshina Dutta

IIT Goa

13^{th} April, 2022

Tutorial-5

Below is a grammar for expressions involving operator + and integer or floating-point operands.

- $E \rightarrow E + T | T$
- $T \rightarrow num.num | num$

Give an SDD to determine the type of each term T and expression E

- $T \rightarrow num.num \; \{ \; T.t = \text{“}float\text{”} \}$
- $T \rightarrow num \; \{ \; T.t = \text{“}int\text{”} \}$
- $E \rightarrow T \; \{ \; E.t = T.t \}$
- $E \rightarrow E + T$
  $\{ \; \text{if}(E.t == \text{“}float\text{”} \; \&\& \; T.t == \text{“}int\text{”}) \; E.t = \text{“}float\text{”}; \cdots \}$

Extend your SDD to translate expressions into postfix notation. Use unary operator **intToFloat** to turn an integer into an equivalent float.

- $E \rightarrow E + T \ \{ \ E.code || T.code || '+' \}$
- $T \rightarrow num \ \{ \ T.code = intToFloat(num) \}$

|| is called concatenation.

Tutorial

Example 5.13: In C, the type **int** [2][3] can be read as, "array of 2 arrays of 3 integers." The corresponding type expression *array*(2, *array*(3, *integer*)) is represented by the tree in Fig. 5.15. The operator *array* takes two parameters, a number and a type. If types are represented by trees, then this operator returns a tree node labeled *array* with two children for a number and a type.

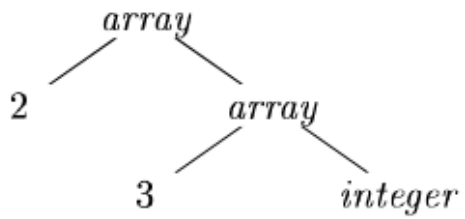


Figure 5.15: Type expression for **int**[2][3]

Tutorial

- T has synthesized attribute t to denote the overall expression
  - The type of $int[2][3]$ is $array(2, array(3, integer))$
- B has a synthesized attribute t to denote the basic type e.g., *int* or *float*
- C has two types
  - $C.b$ is an inherited attribute to pass the basic type down the tree i.e., till the end of the expression
  - $C.t$ is a synthesized attribute to accumulate the type information from the bottom to the top

Tutorial

Actually,
C and C1 are
same. Just
to avoid confusion
we kept diff
names.

PRODUCTION	SEMANTIC RULES
$T \rightarrow B\ C$	$T.t = C.t$
	$C.b = B.t$
$B \rightarrow \mathbf{int}$	$B.t = integer$
$B \rightarrow \mathbf{float}$	$B.t = float$
$C \rightarrow [\ \mathbf{num}\ ]\ C_1$	$C.t = array\,(\mathbf{num}.val,\ C_1.t)$
	$C_1.b = C.b$
$C \rightarrow \epsilon$	$C.t = C.b$

Figure 5.16: T generates either a basic type or an array type

Tutorial

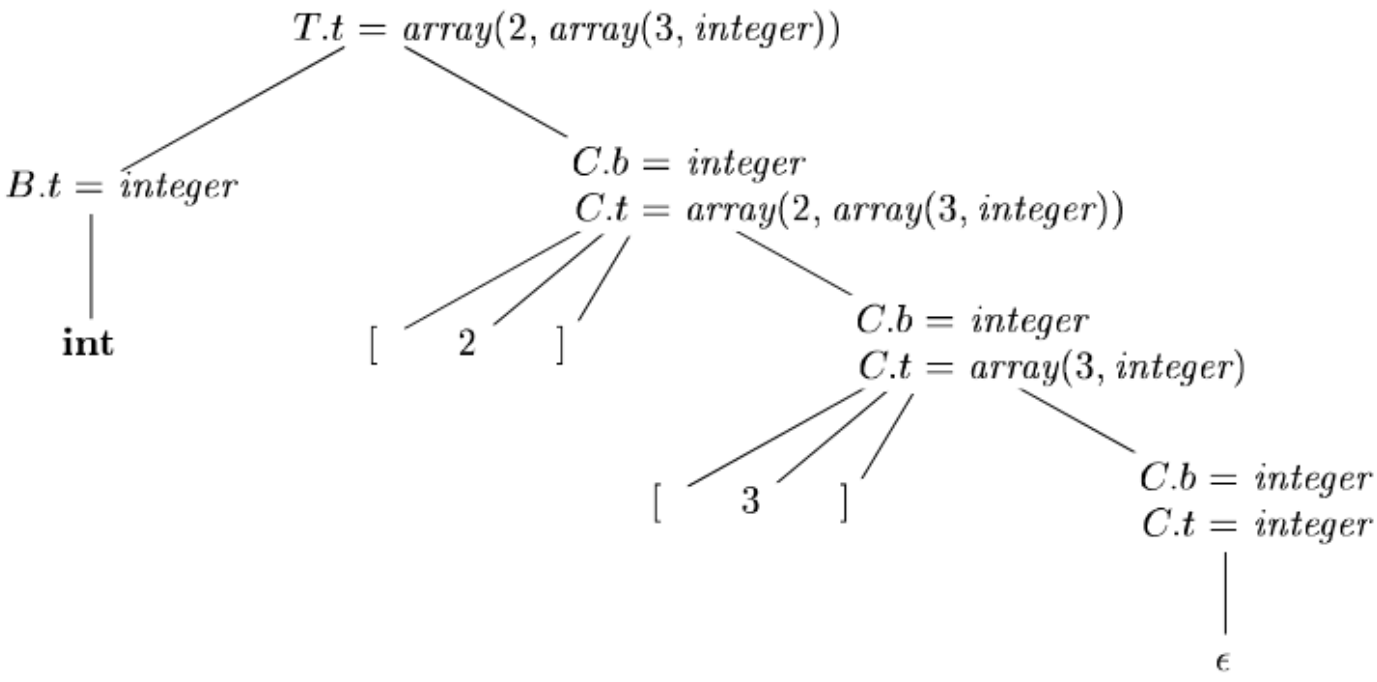
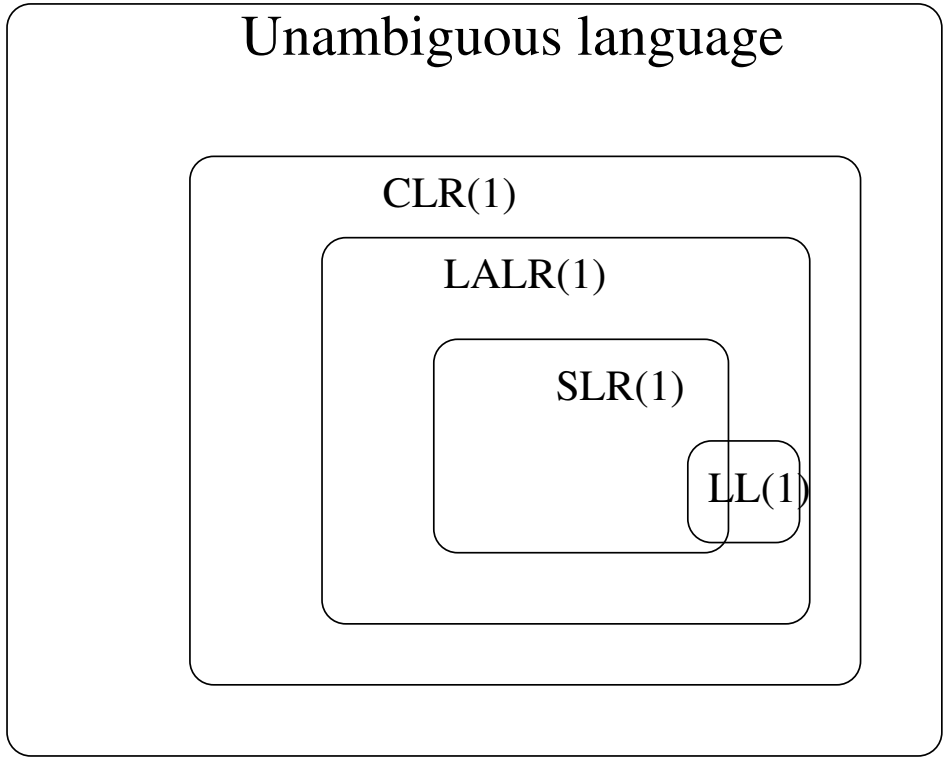


Figure 5.17: Syntax-directed translation of array types

Parsing techniques



Inherently Ambiguous Language

Let L be a Context Free Language (CFL). If every Context-Free Grammar G with Language $L = L(G)$ is ambiguous, then L is said to be inherently ambiguous Language.

- $\{a^n b^m c^m\} \cup \{a^n b^n c^m\}$

Inherently Ambiguous Language

- $S \rightarrow A|B$
- $A \rightarrow aA|C$
- $C \rightarrow bCc|\epsilon$
- $B \rightarrow Bc|D$
- $D \rightarrow aDb|\epsilon$

If for the given Language L,
All the grammars that can accept L are ambiguous, we say L is inherently ambiguous language.