



# Compiler Design

---

**Preet Kanwal**

Department of Computer Science & Engineering

Teaching Assistant : Kavya P K

# Compiler Design

---

## Unit 3: Syntax Directed Definitions

**Preet Kanwal**

Department of Computer Science & Engineering

In this lecture, you will learn about -

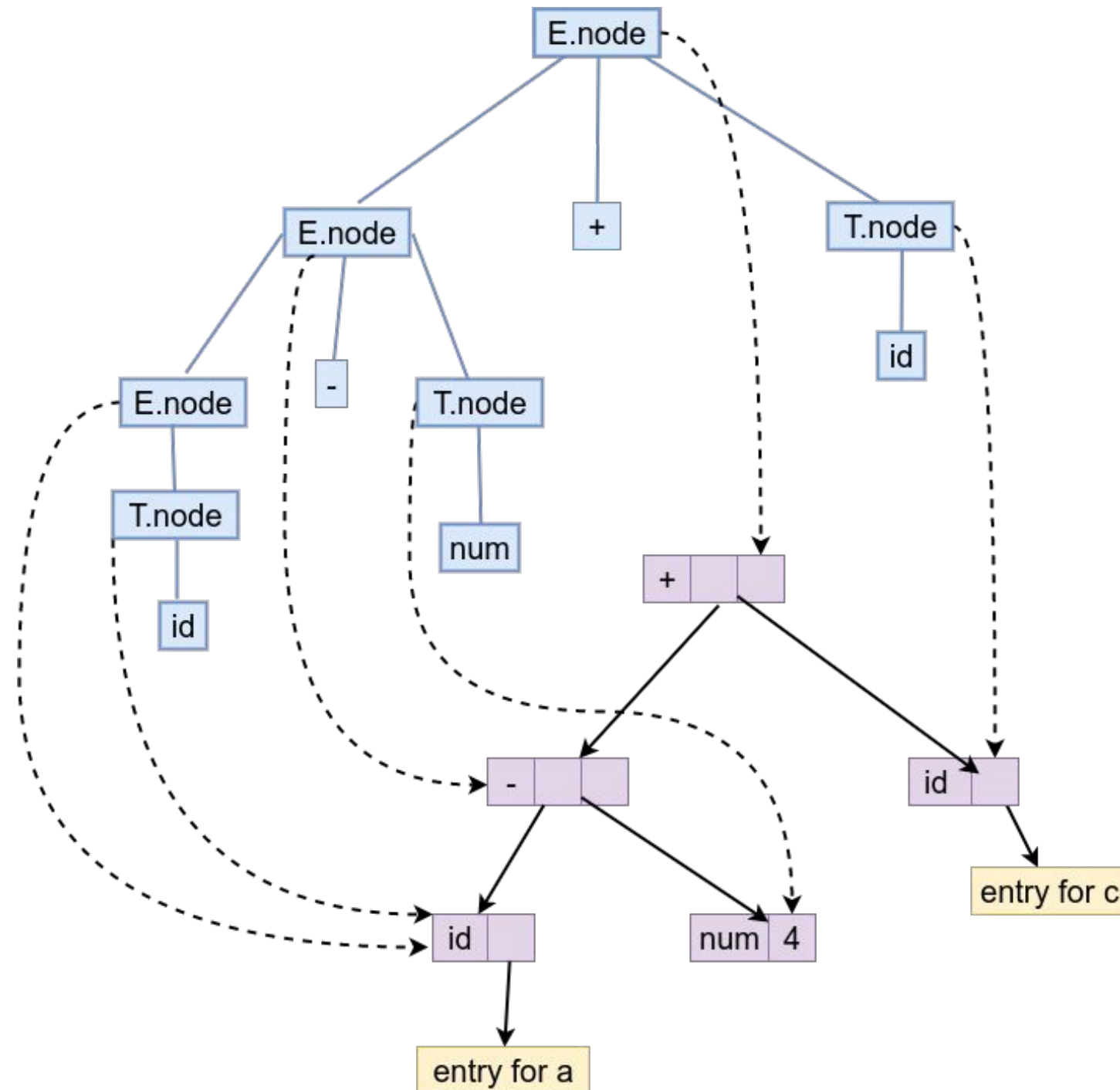
- **S-Attributed SDD Examples :**
  - To generate Syntax tree for Expressions
  - To generate Syntax tree for Statements

Example 1 - SDD to generate Syntax tree for Expressions

Production	Semantic Rule
$E \rightarrow E_1 + T$	$\{ E.node = new Node( '+' , E_1.node, T.node); \}$
$E \rightarrow E_1 - T$	$\{ E.node = new Node( '-' , E_1.node, T.node); \}$
$E \rightarrow T$	$\{ E.node = T.node ; \}$
$T \rightarrow ( E )$	$\{ T.node = E.node ; \}$
$T \rightarrow id$	$\{ T.node = new Leaf( id , id.entry); \}$
$T \rightarrow num$	$\{ T.node = new Leaf( num , num.lexval); \}$

## Example 1 - SDD to generate Syntax tree for Expressions

Use the previous grammar to construct the syntax tree for the input **a - 4 + c**



## Example 2 - SDD to generate Syntax tree for Statements

Production	Semantic Rule
<b>Stmt -&gt; S Stmt</b>	<i>{ Stmt.node = new Node(Seq, S.node, Stmt.node); }</i>
<b>Stmt -&gt; S</b>	<i>{ Stmt.node = S.node; }</i>
<b>S -&gt; if (cond) { Stmt }</b>	<i>{ S.node = new Node(if, Cond.node, Stmt.node); }</i>
<b>S -&gt; while (cond) { Stmt }</b>	<i>{ S.node = new Node(while, Cond.node, Stmt.node); }</i>
<b>S -&gt; AssignExpr</b>	<i>{ S.node = AssignExpr.node ; }</i>
<b>Cond -&gt; E<sub>1</sub> &gt; E<sub>2</sub></b>	<i>{ Cond.node = new Node( &gt;, E<sub>1</sub>.node, E<sub>2</sub>.node); }</i>
<b>Cond -&gt; E<sub>1</sub> &lt; E<sub>2</sub></b>	<i>{ Cond.node = new Node( &lt;, E<sub>1</sub>.node, E<sub>2</sub>.node); }</i>
<b>Cond -&gt; E<sub>1</sub>    E<sub>2</sub></b>	<i>{ Cond.node = new Node(   , E<sub>1</sub>.node, E<sub>2</sub>.node); }</i>
<b>Cond -&gt; E<sub>1</sub> &amp;&amp; E<sub>2</sub></b>	<i>{ Cond.node = new Node(&amp;&amp;, E<sub>1</sub>.node, E<sub>2</sub>.node); }</i>
<b>AssignExpr -&gt; id = E;</b>	<i>{ AssignExpr.node = new Node(=, new Leaf(id,id.entry), E.node); }</i>

Example 2 - SDD to generate Syntax tree for Statements

Production	Semantic Rule
...	...
$E \rightarrow E_1 + T$	$\{ E.node = new Node( '+' , E_1.node, T.node); \}$
$E \rightarrow T$	$\{ E.node = T.node; \}$
$T \rightarrow T_1 * F$	$\{ T.node = new Node( '*' , T_1.node, F.node); \}$
$T \rightarrow F$	$\{ T.node = F.node; \}$
$F \rightarrow id$	$\{ F.node = new Leaf( id , id.entry); \}$
$F \rightarrow num$	$\{ F.node = new Leaf( num , num.lexval); \}$

Use the previous grammar to construct the syntax tree for the input

The diagram illustrates the mapping from an abstract syntax tree (AST) to a control flow graph (CFG) for the code snippet: `if (x > 10) x = 10;`

**Abstract Syntax Tree (AST):**

- if.node**
  - cond.node**
    - E.node** (T.node, id)
    - >**
    - E.node** (T.node, F.node, num)
  - Stmt.node** (S.node, AssignExpr.node)
    - AssignExpr.node**
      - id**
      - =**
      - E.node** (T.node, F.node, num)

**Control Flow Graph (CFG):**

- if** (entry node)
  - >** (branch node)
    - id** (exit node) → **entry for x**
    - num 10** (exit node)
  - =** (branch node)
    - id** (exit node) → **entry for x**
    - num 10** (exit node)

**Mapping (Dashed Arrows):**

- if.node** maps to the **if** node in the CFG.
- cond.node** maps to the **>** node in the CFG.
- AssignExpr.node** maps to the **=** node in the CFG.
- E.node** (under **cond.node**) maps to the **id** node in the CFG.
- F.node** (under **cond.node**) maps to the **num 10** node in the CFG.
- id** (under **AssignExpr.node**) maps to the **id** node in the CFG.
- =** (under **AssignExpr.node**) maps to the **=** node in the CFG.
- E.node** (under **AssignExpr.node**) maps to the **num 10** node in the CFG.





**THANK  
YOU**

---

**Preet Kanwal**

Department of Computer Science & Engineering

[preetkanwal@pes.edu](mailto:preetkanwal@pes.edu)