# CS & IT ENGINEERING

## Compiler Design

Syntax Directed Translations

Lecture No. 1

By- DEVA Sir


→ What is SDT?

→ Attributes

→ Definitions

# Semantic Analysis Vs SDT

- It uses SDT to perform type checking, function compatibility, variable declaration, ...

- It is more powerful program or tool that can be used for
  1) Semantic Analysis
  2) parse tree Generation
  3) Intermediate code
  4) Evaluations of Exps
  5) conversions

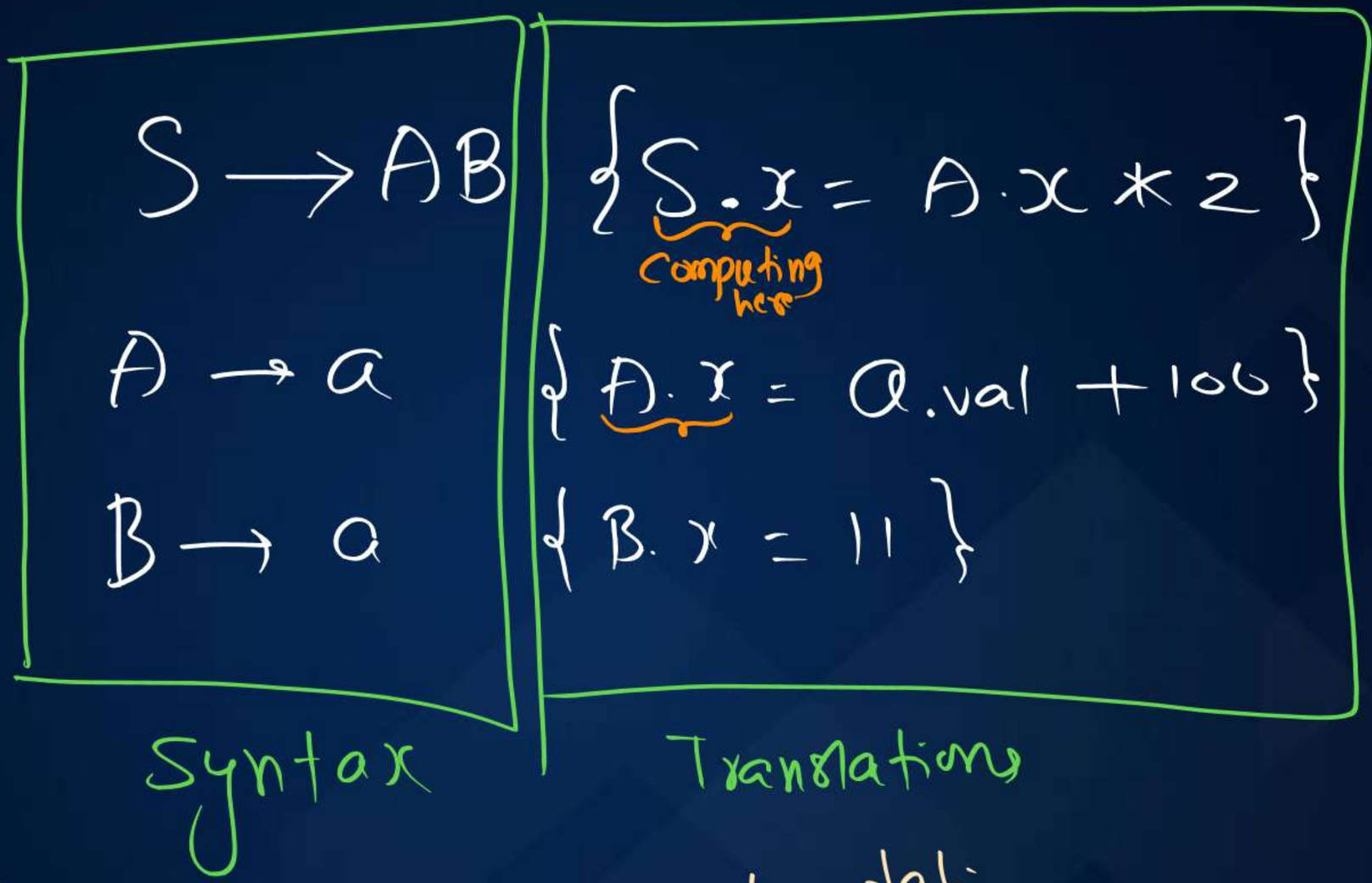
# Syntax Directed Translation (SDT)

SDT = syntax + Translation

(Structure)

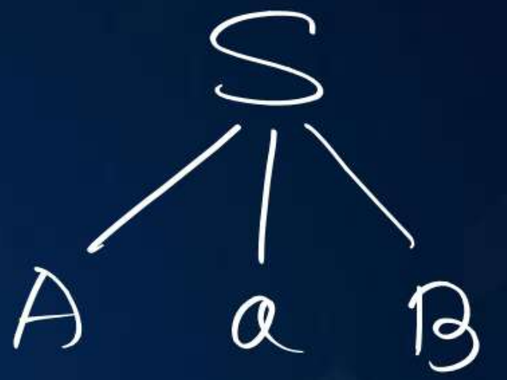
- → Semantic
- → Evaluation
- → Actions
- → Specific requirement
- → program

= CFG + Translations {. . . .}

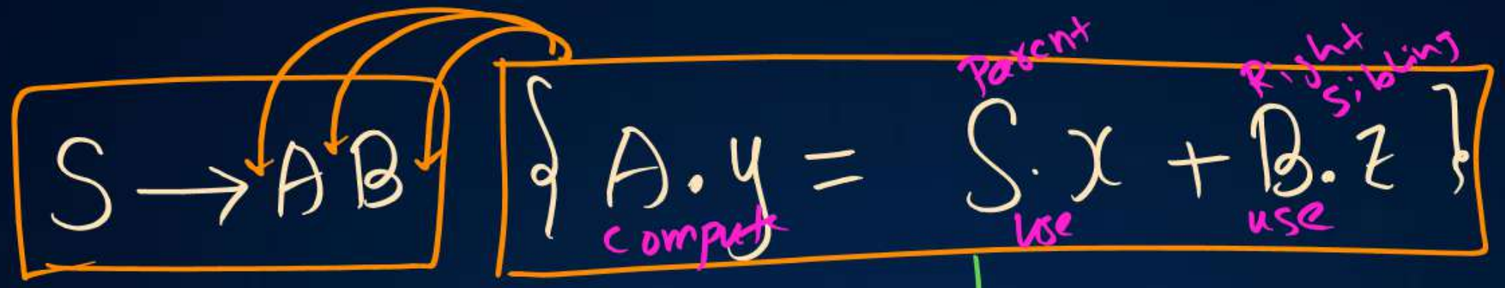



Every production has zero or more translations

$S \longrightarrow AaB$

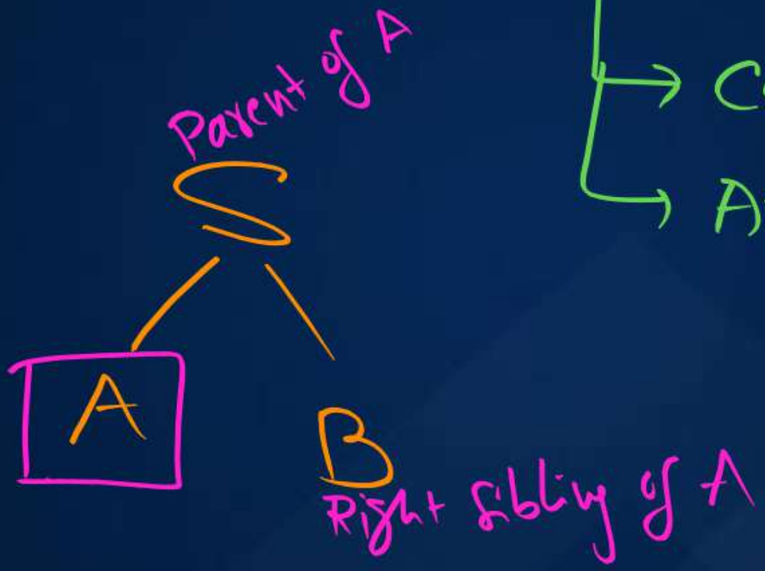



Parent / siblings / child

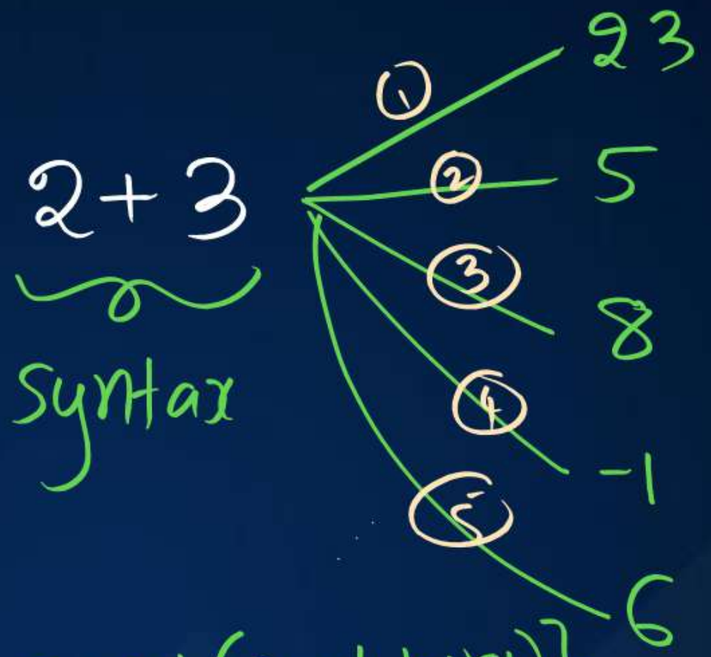



→ Computation

→ Attributes




x
y
z
} attributes




$c = a + b$

① { c.val = concat(a.val, b.val) }

② { c.val = a.val + b.val }

③ { c.val = pow(a.val, b.val) }

④ { c.val = a.val - b.val }

⑤ { c.val = a.val * b.val }

Attribute

- type
- lex
- val
- num
- x
- y
- s
- i
- any

A.type

A.val

A.x

# Attributes

→ ① Inherited Attribute

$S \rightarrow A\ a\ \boxed{B}$

$\{B.val = S.val * A.val\}$

Computation

computation depends on parent/siblings

② Synthesized Attribute

$\boxed{S} \rightarrow A\ a\ B$

$\{S.val = A.val + a.val\}$

Computation

computation depends on children

Find Attribute type

① $S \rightarrow a \quad \{ S.x = a.val \}$

~~val~~ (we are only using)

x is synthesized

② $S \rightarrow S_1 a \quad \{ S_1.x = S.x + 1 \; ; \; S.y = S_1.y * 2 \}$

$S \rightarrow b \quad \{ S.x = b.val - 1 \; ; \; S.y = S.x + 9 \}$

inherited

synthesized

Synthesized

we can use anywhere

x

y

~~val~~

x is neither inherited nor synthesized

y is synthesized

③ $E \longrightarrow E_1 + E_2 \quad \{E.x = E_1.x + E_2.x\}$

$E \longrightarrow a \quad \{E.x = a.val\}$

$x$ is Synthesized attribute

(4)

$D \rightarrow T\ I\ ;$ { I.type = T.type } (Inherited)

$T \rightarrow int$ { T.type = int } (Synthesized)

$I \rightarrow a$ { }

type is ~~neither inherited nor synthesized~~ Attribute

# SDTs

## Specifications

for writing SDTs

by using "Definitions"

## Evaluations

for translating SDTs

# Definitions of SDTs :

- ① L-attributed Grammar — [L-attributed SDT] (L-attributed Definition)
- ② S-attributed Grammar — [S-attributed SDT] (S-attributed Definition)

Based on "Computation" and "Position of Translation"

## ① L-attributed SDT

I) Computation depends on

Parent / Left siblings / Children

$S \rightarrow a \quad \{S.val = a.num\}$

$S \rightarrow AB \quad \{B.val = S.val\}$
$\{A.val = S.val\}$
$\{B.x = S.x + A.y\}$
$\{S.x = A.x + B.x\}$

Note : Except right sibling, it can depend on any.

II) translations can appear anywhere

$S \rightarrow \{\ldots\}\, a$

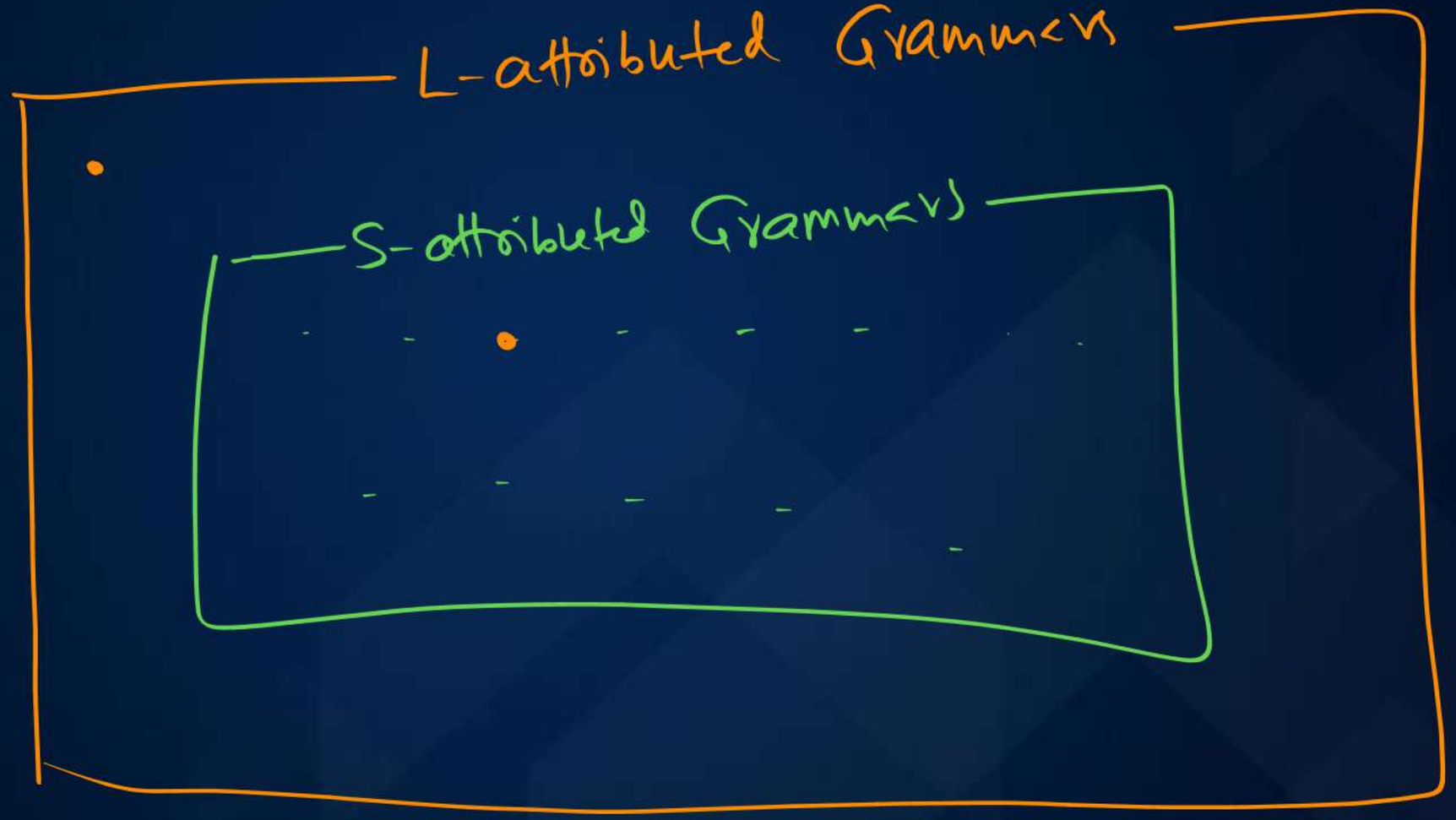
## S-attributed SDT

I) Computation depends on only Children
[It uses Synthesized Attributes]

II) Translation must appear only at end of production

$S \rightarrow a \{\ldots\}$

Every S-attributed SDT is always L-attributed.



(1)

$D \rightarrow T\ I\ ;\ \{ I.type = T.type \}$

$T \rightarrow int\ \{ T.type = int \}$

$I \rightarrow a\ \{\ \}$

→ This SDT is L-attributed but not S-attributed

(2)

$E \rightarrow E_1 + E_2 \quad \{ E.val = E_1.val + E_2.val \}$

$E \rightarrow a \quad \{ E.val = a.num \}$

↳ SDT is S-attributed & L-attributed

(3)

$E \rightarrow E_1 \{E.val = E_1.val + E_2.val\} + E_2$ ;

not at the end

$E \rightarrow a \quad \{E.val = a.num\}$

→ SDT is not S-attributed

is L-attributed

Take all SDTs from GATE PYQ.

Q1) Find attribute type

Q2) Find SDT Definition

## Summary

- SDT ✓
- Attributes ✓
- Definitions ✓

Next: Evaluation ?


THANK YOU
GW
SOLDIERS !