

Syntax Directed Translation

Syntax directed translation

- Yacc can do a simple kind of syntax directed translation from an input sentence to C code
- We can also think of it as compilation
- Each node in a parse tree produces a value
 - That value depends on the type of the node
 - And on the values produced by its children
- The value is usually produced by a *rule* associated with the node
- This is just the rules in Yacc, e.g.:
 - { \$\$ = \$1 + \$3; }

Why is a 200 after this?

```
pcalc> calc  
331 Calculator  
(type ? for help and . to exit)
```

```
>> = a 1  
1
```

```
>> if 1 (= a 100) (= a 200)  
100
```

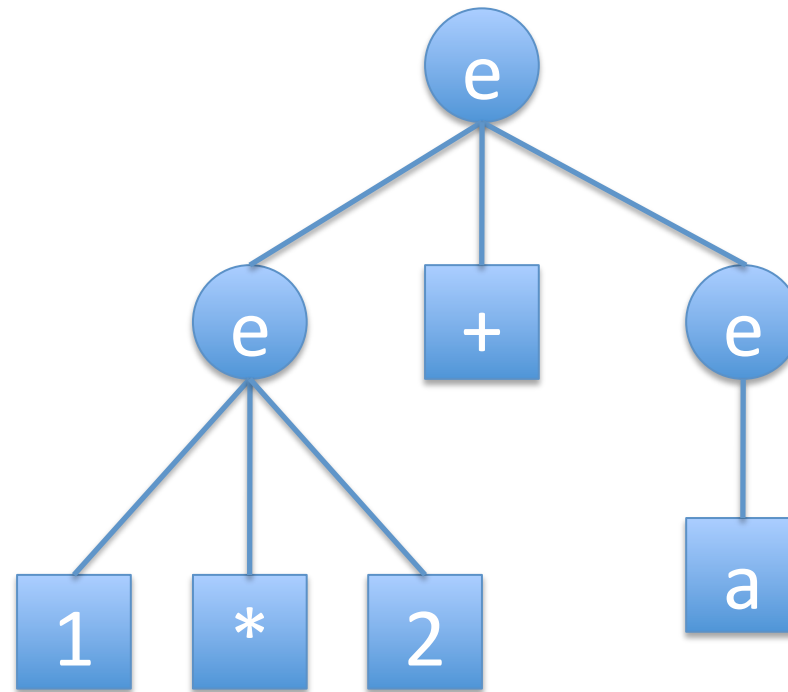
```
>> a  
200
```

```
>>
```

Example

```
e: e '+' e    {$$ = $1+$3;}  
  | e '*' e   {$$ = $1*$3;}  
  | NAME     {$$ = $1->value;}  
  | NUMBER  {$$ = $1;}
```

1 * 2 + a

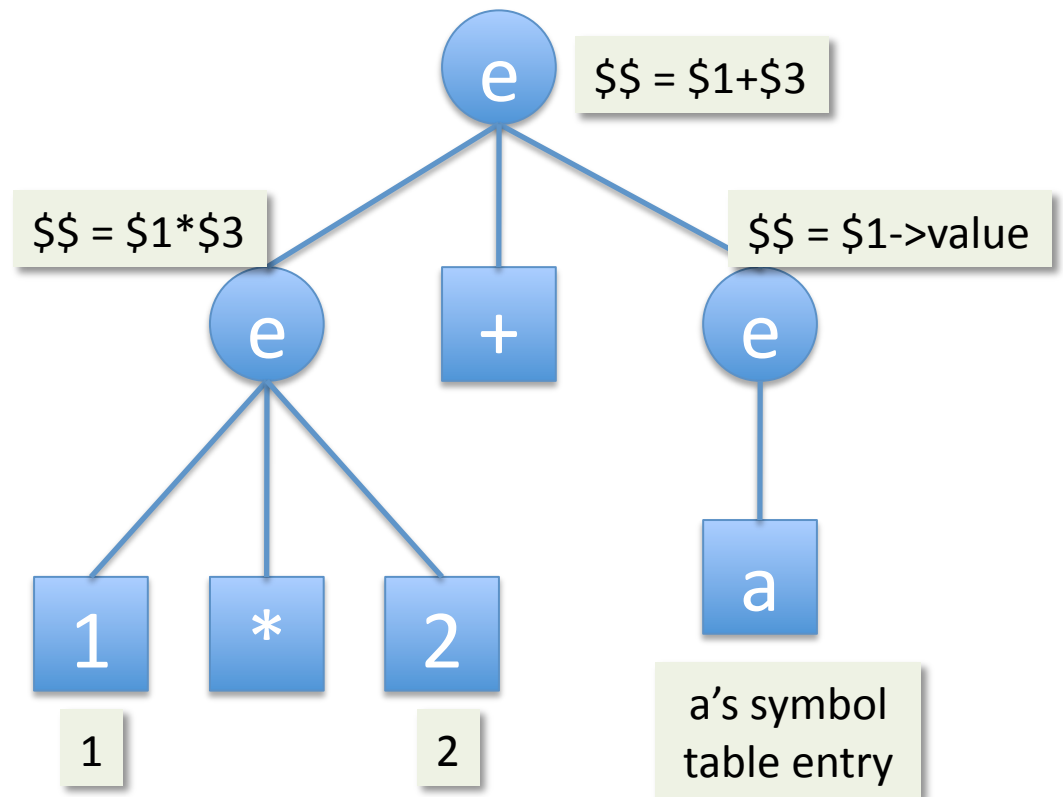


grammar + input string => parse tree

Example

```
e: e '+' e    {$$ = $1+$3;}  
  | e '*' e   {$$ = $1*$3;}  
  | NAME     {$$ = $1->value;}  
  | NUMBER  {$$ = $1;}
```

1 * 2 + a

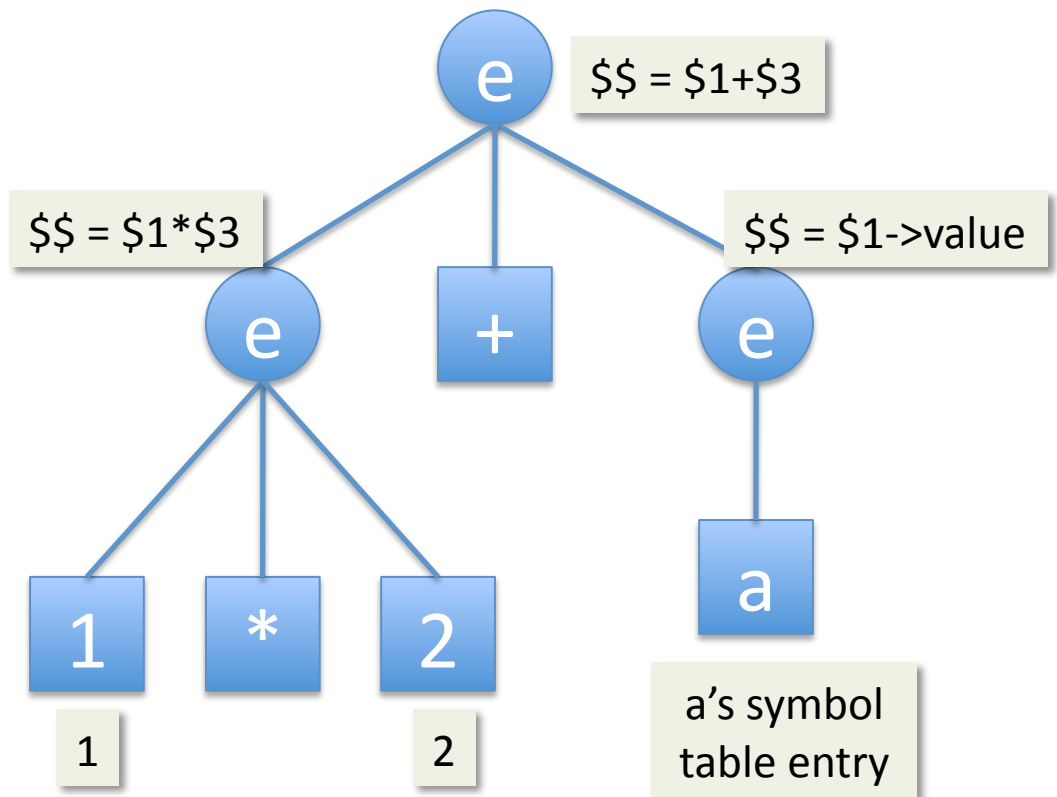


grammar + input string => parse tree + annotations

Example

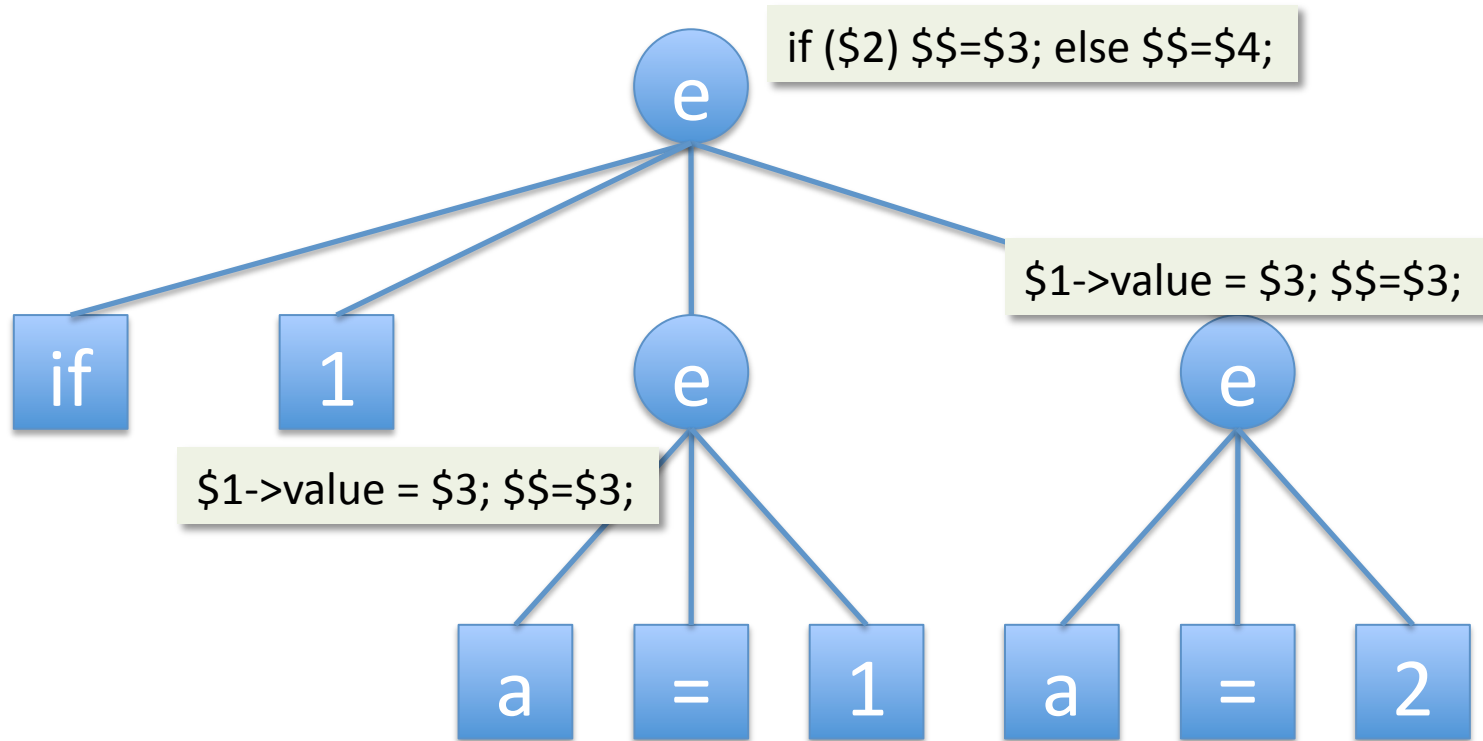
```
e: e '+' e    {$$ = $1+$3;}  
  | e '*' e   {$$ = $1*$3;}  
  | NAME     {$$ = $1->value;}  
  | NUMBER  {$$ = $1;}
```

1 * 2 + a



Do a post order traversal of the annotated parse tree to determine the execution order of the nodes.

Conditionals



- If evaluates all three args and selects the value to return
- Evaluation is bottom up, left to right
- Watch out for side effects!