Compiladores – ASTs

Fabio Mascarenhas – 2017.1

http://www.dcc.ufrj.br/~fabiom/comp

Árvores Sintáticas Abstratas (ASTs)

- A árvore de análise sintática tem muita informação redundante
  - Separadores, terminadores, não-terminais auxiliares (introduzidos para contornar limitações das técnicas de análise sintática)
- Ela também trata todos os nós de forma homogênea, dificultando processamento deles
- A árvore sintática abstrata joga fora a informação redundante, e classifica os nós de acordo com o papel que eles têm na estrutura sintática da linguagem
- Fornecem ao compilador uma representação compacta e fácil de trabalhar da estrutura dos programas

Exemplo

- Seja a gramática abaixo:

```
E -> n
  | ( E )
  | E + E
```

- E a entrada `25 + (42 + 10)`

- Após a análise léxica, temos a sequência de tokens (com os lexemes entre parênteses):

n(25) ‘+’ ‘(‘ n(42) ‘+’ n(10) ‘)’

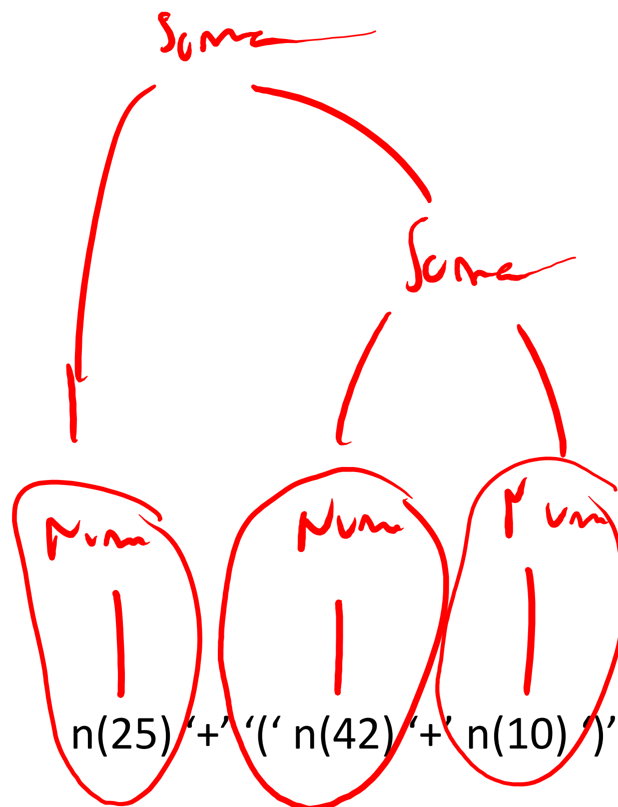
- Um analisador sintático bottom-up construiria a árvore sintática da próxima página

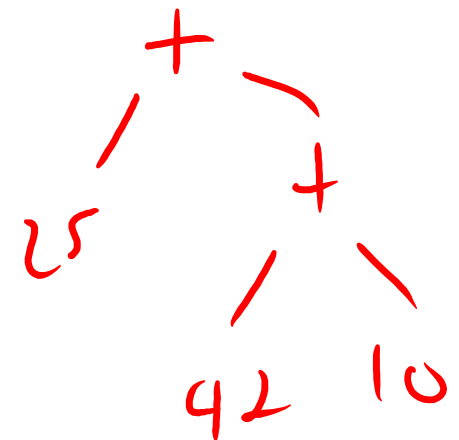
Exemplo – árvore sintática

E -> n
| (E)
| E + E

Exemplo - AST

E -> n
  | (E)
  | E + E





Representando ASTs

- Cada estrutura sintática da linguagem, normalmente dada pelas produções de sua gramática, dá um tipo de nó da AST
- Em um compilador escrito em Java, vamos usar uma classe para cada tipo de nó
- Não-terminais com várias produções ganham uma interface ou uma classe abstrata, derivada pelas classes de suas produções
- Nem toda produção ganha sua própria classe, algumas podem ser redundantes

```
E -> n          => Num (deriva de Exp)
   | ( E )      => Redundante
   | E + E      => Soma (deriva de Exp)
```

Exemplo – Representando a AST

```
interface Exp {}

class Num implements Exp {
  int val;

  Num(String lexeme) {
    val = Integer.parseInt(lexeme);
  }
}

class Soma implements Exp {
  Exp e1;
  Exp e2;

  Soma(Exp _e1, Exp _e2) {
    e1 = _e1; e2 = _e2;
  }
}
```

Uma AST para TINY

- Vamos lembrar da gramática SLR de TINY:

```
TINY -> CMDS
CMDS -> CMDS ; CMD
      | CMD
CMD  -> if EXP then CMDS end
      | if EXP then CMDS else CMDS end
      | repeat CMDS until EXP
      | id := EXP
      | read id
      | write EXP
```

```
EXP -> EXP < EXP
     | EXP = EXP
     | EXP + EXP
     | EXP - EXP
     | EXP * EXP
     | EXP / EXP
     | ( EXP )
     | num
     | id
```

- Vamos representar listas (CMDS) usando a própria interface `List<T>` de Java

Uma AST para TINY - Resumo

- Duas interfaces: `Cmd`, `Exp`
- As duas produções do `if` compartilham o mesmo tipo de nó da AST
- Quatorze classes concretas
- Poderíamos juntar todas as operações binárias em uma única classe, e fazer a operação ser mais um campo
- Ou poderíamos ter separado `If` e `IfElse`
- Não existe uma maneira certa; a estrutura da AST é engenharia de software, não matemática