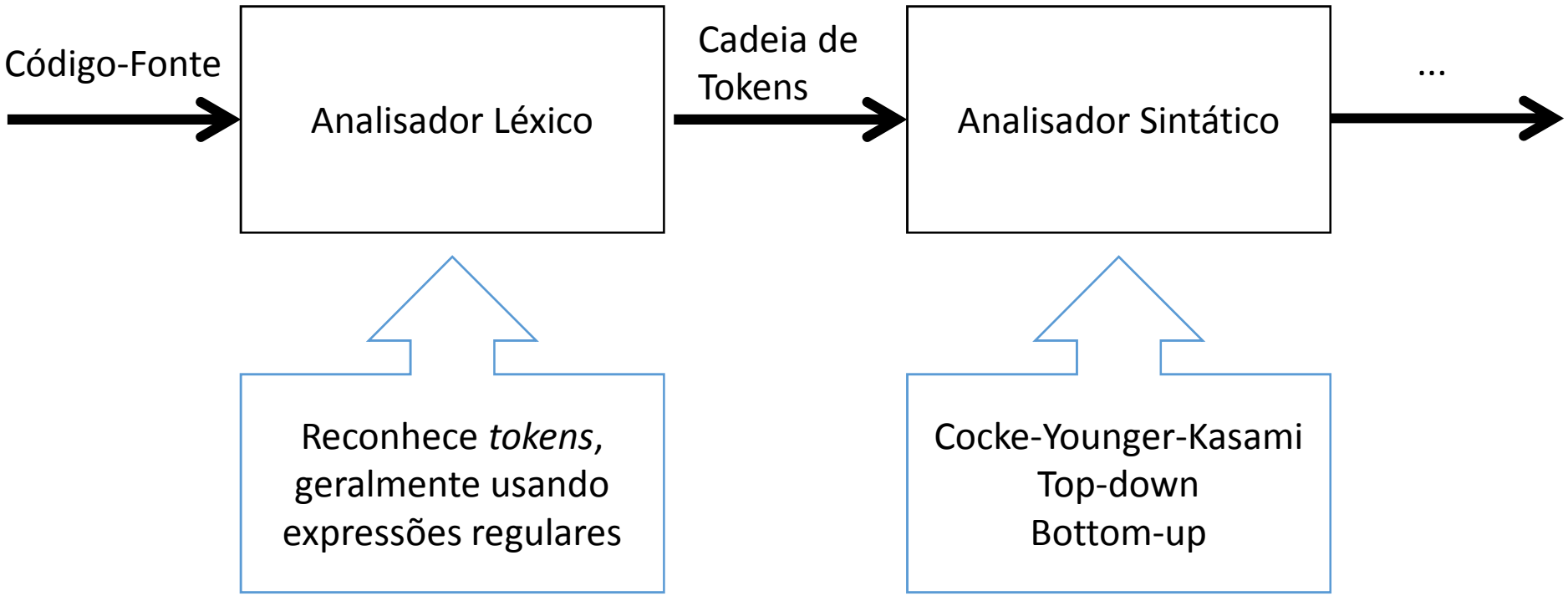
Teoria da Computação - Seminário 2
Linguagens Livres de Contexto

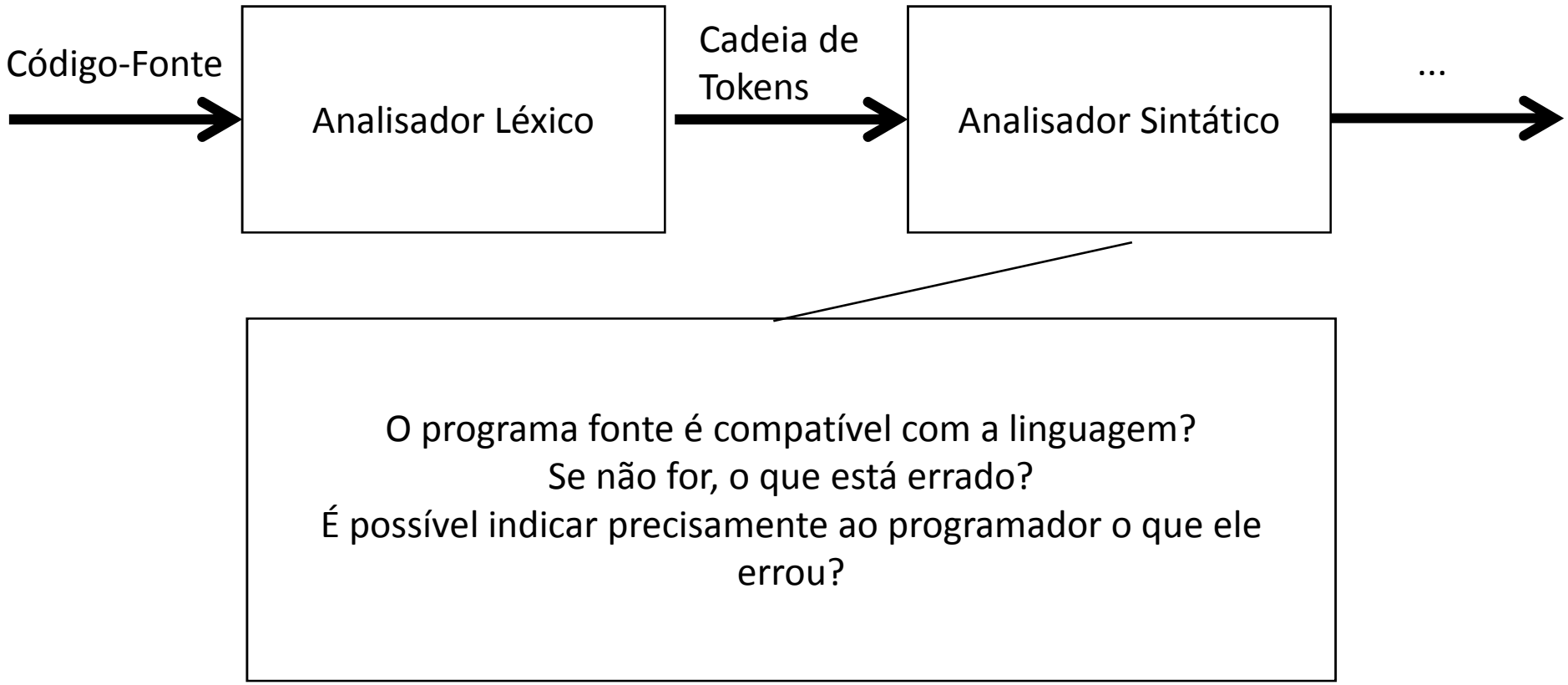
Análise Sintática de Descendência Recursiva

Fernanda Borges da Silva
Cristiano Oliveira Gonçalves

Introdução | Panorama geral de compiladores



Introdução | Panorama geral de compiladores



Introdução | Análise Sintática top-down

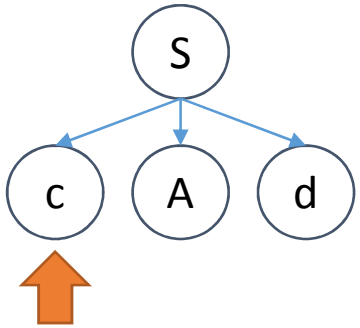
- Também chamada de Análise Sintática Descendente
- Tentativa de se encontrar uma derivação mais à esquerda
- Tentativa de se construir uma árvore gramatical para a cadeia de entrada
- Constrói da raiz para as folhas
- Há duas formas de analisadores sintáticos descendentes: que pode envolver retrocesso (recursivos) e sem retrocesso (preditivos)

Análise Sintática de Descendência Recursiva

Exemplo no qual o retrocesso é exigido

S → cAd
A → ab | a

Cadeia w = cad



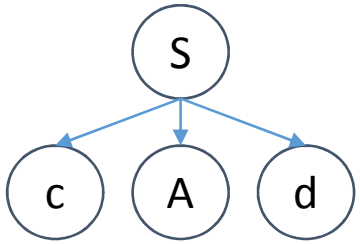
Análise Sintática de Descendência Recursiva

Exemplo no qual o retrocesso é exigido

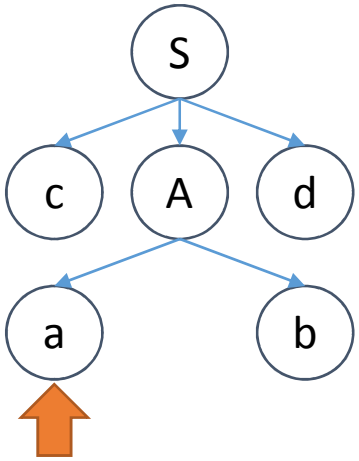
S → cAd
A → ab | a

Cadeia w = cad

(a)



(b)



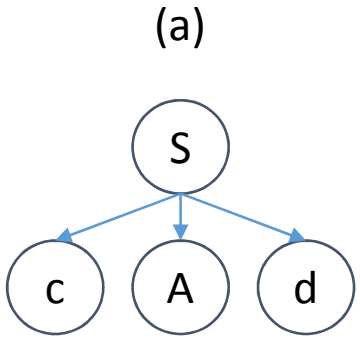
Tentar a primeira alternativa de A

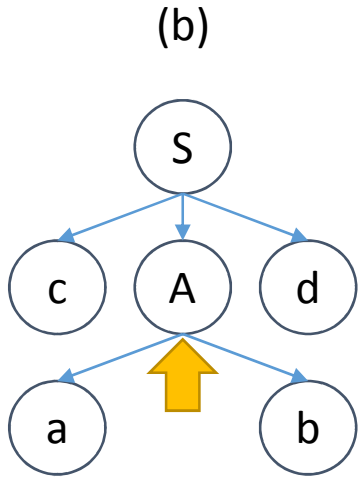
Análise Sintática de Descendência Recursiva

Exemplo no qual o retrocesso é exigido

S → cAd
A → ab | a

Cadeia w = cad





Análise Sintática de Descendência Recursiva

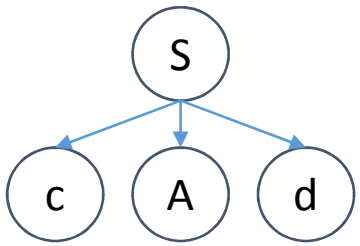
Exemplo no qual o retrocesso é exigido

S → cAd
A → ab | a

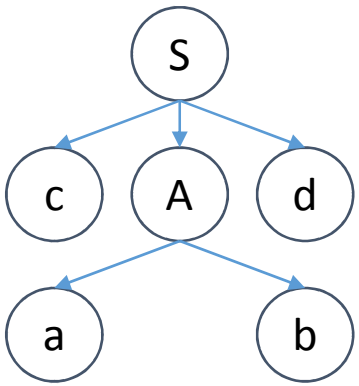
Cadeia w = cad

Ao irmos de volta para A precisamos restabelecer o apontador da entrada para a posição 2

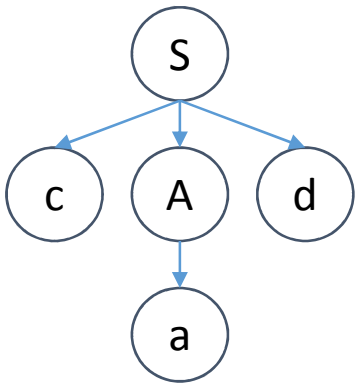
(a)



(b)



(c)



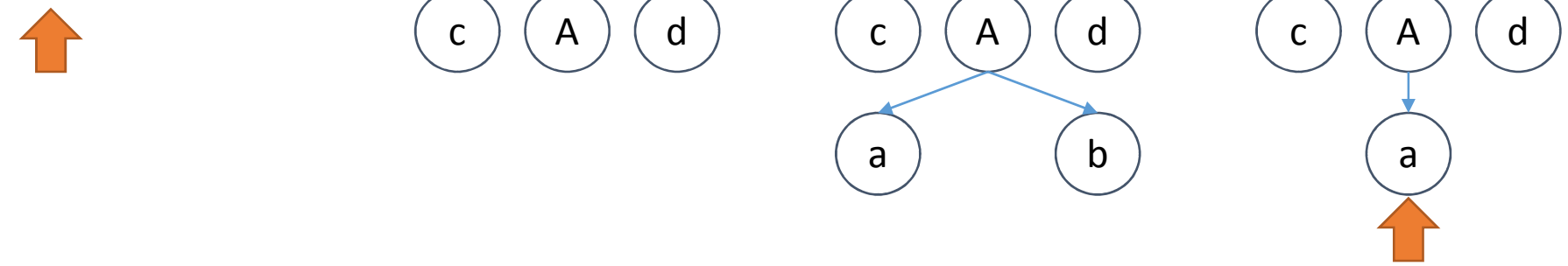
Tentar a segunda alternativa de A

Análise Sintática de Descendência Recursiva

Exemplo no qual o retrocesso é exigido

S → cAd
A → ab | a

Cadeia w = cad

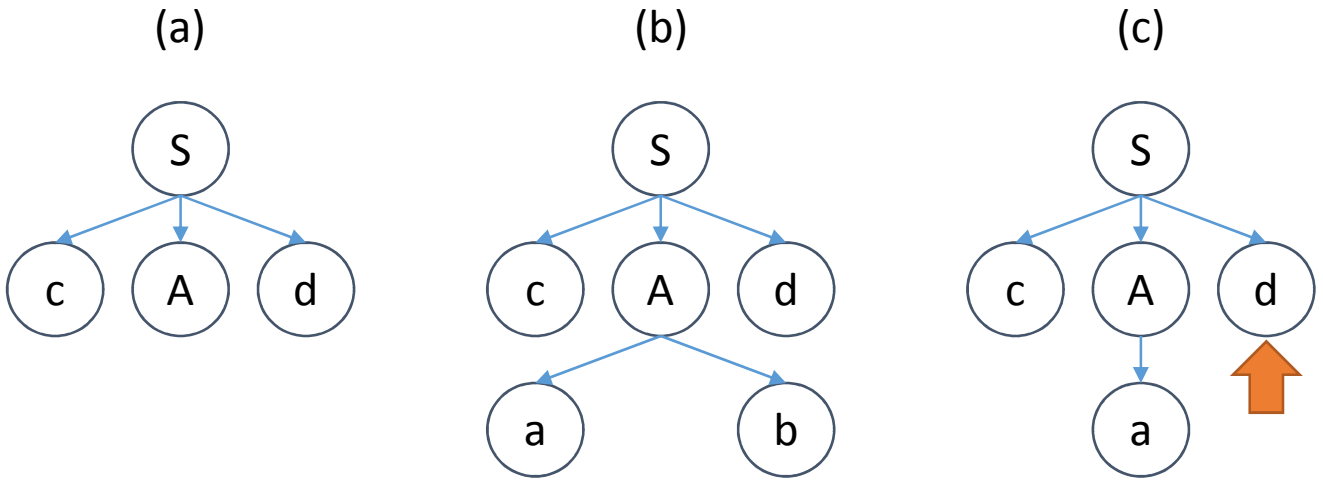


Análise Sintática de Descendência Recursiva

Exemplo no qual o retrocesso é exigido

S → cAd
A → ab | a

Cadeia w = cad

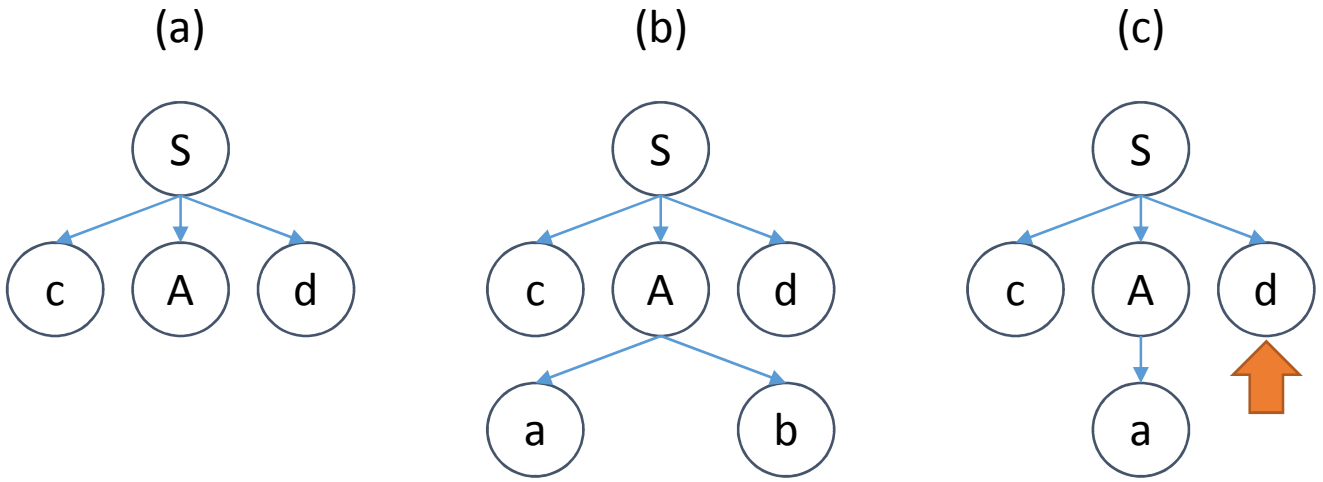


Análise Sintática de Descendência Recursiva

Exemplo no qual o retrocesso é exigido

S → cAd
A → ab | a

Cadeia w = cad

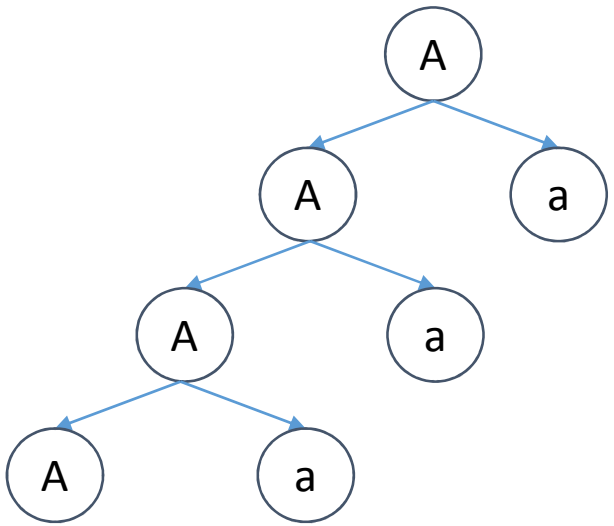


O problema com recursões à esquerda

ba*: A -> Aa|b

Recursões à esquerda impedem o analisador de chegar em estados finais e prosseguir para o próximo passo!

baaa:



...

Como eliminar o problema?

ba*: A -> bA’, A’ -> aA’|ε

Análise Sintática de Descendência Recursiva

Exemplo de código para a gramática E → T | T + E, T → a | a * T | (E)

```
var cursor = 0;

var savedCursor = cursor;

function E() {
  return (saveCursor(), E1()) || (backtrack(), saveCursor(), E2());
}

function E1() {
  return T();
}

function E2() {
  return T() && term('+') && E();
}

function T() {
  return (saveCursor(), T1()) ||
    (backtrack(), saveCursor(), T2()) ||
    (backtrack(), saveCursor(), T3());
}

function T1() {
  return term('a');
}

function T2() {
  return term('a') && term('*') && T();
}

function T3() {
  return term('(') && E() && term(')');
}
```

```
function saveCursor() {
  savedCursor = cursor;
}

function backtrack() {
  cursor = savedCursor;
}

function term(expected) {
  return getNextToken() === expected;
}

function getNextToken() {
  // Skip whitespace.
  while (source[cursor] === ' ') cursor++;
  var nextToken = source[cursor];
  cursor++;
  return nextToken;
}

var source;

function parse(s) {
  source = s;
  cursor = 0;
  // We succeed if our main E symbol succeeds *and* we parse
  // all tokens (reached end of the source string).
  return E() && cursor == source.length;
}

// Test time!
console.log('(a)', parse('(a)')); // true
console.log('-a', parse('-a')); // false
```

Conclusão

- A descendência recursiva com retrocesso não é o método mais eficiente de análise sintática

- Muito útil quando não for possível gerar gramáticas que não necessitem de retrocesso

- O estudo ajudou a esclarecer a relação entre teoria da computação e compiladores, cuja teoria fundamenta o uso generalizado de computadores

Referências Bibliográficas

- Compilers: Principles, Techniques, and Tools (2nd Edition). Alfred Aho, Monica Lam, Ravi Sethi, and Jeffrey Ullman. Addison-Wesley, 2006

- https://gist.github.com/DmitrySoshnikov/e2c3a793636dc03f2200

- https://www.tutorialspoint.com/compiler_design/compiler_design_lexical_analysis.htm

- http://comp.ist.utl.pt/aaa/Prog/Compiladores%20-%20Principios%20Tecnicas%20E%20Ferramentas%20(Pt%20Br).pdf

- Página 81

- https://gist.github.com/DmitrySoshnikov/e2c3a793636dc03f2200