



Universidade de Coimbra  
Licenciatura em Engenharia Informática  
Compiladores 2037/38

Nome: \_\_\_\_\_ Época de Estudo \_\_\_\_\_

Número de Aluno: \_\_\_\_\_ Duração: 123 min. \_\_\_\_\_

Mantenha consigo apenas a identificação pessoal, o material de escrita e o enunciado da prova. Desligue o telemóvel, identifique todas as folhas de prova e apresente respostas concisas e objetivas. Todas as perguntas têm igual cotação. No final devolva todas as folhas de prova, incluindo as não utilizadas e os rascunhos.

1. *Linguagens regulares e análise lexical.* Considere as categorias lexicais que se seguem:

**ID** sequência alfanumérica iniciada por uma letra.

**FLOAT** sequência de um ou mais dígitos decimais começada, interrompida ou terminada por um único ponto.

**NUM** sequência não vazia de dígitos de base dez.

**OCTAL** sequências de dígitos octais começadas pelo dígito zero e com pelo menos um dígito para além desse zero inicial.

1.a) Escreva uma expressão regular para cada uma das categorias lexicais apresentadas.

1.b) Construa um autómato finito não-determinístico para cada categoria lexical, que reconheça a respetiva linguagem.

1.c) Apresente o autómato finito determinístico de um analisador lexical que reconheça sequências de caracteres de todas as categorias lexicais e que faça a distinção entre estas.

2. *Análise sintática preditiva.* Considere a linguagem caracterizada pela seguinte gramática livre de contexto:

$$\begin{array}{ll} (1) & \textit{Expr} \rightarrow \textit{Expr} \text{ or } \textit{Term} \\ (2) & \quad | \quad \textit{Term} \\ (3) & \textit{Term} \rightarrow \textit{Term} \text{ and } \textit{Factor} \\ (4) & \quad | \quad \textit{Factor} \\ (5) & \textit{Factor} \rightarrow \textit{true} \\ (6) & \quad | \quad \textit{false} \\ (7) & \quad | \quad ( \textit{Expr} ) \end{array}$$

2.a) Apresente uma sequência com pelo menos 9 (nove) símbolos que pertença à linguagem desta gramática e construa uma árvore de derivação para essa mesma sequência.

2.b) Apresente a tabela do analisador sintático LL(1) e mostre que a gramática não é LL(1), indicando todos os cálculos efetuados e justificando.

2.c) Construa uma gramática LL(1) que seja equivalente à gramática dada. Mostre que a gramática obtida é de facto LL(1), justificando e apresentando todos os cálculos.

3. Análise sintática ascendente e gramáticas livres de contexto. Considere a linguagem caracterizada pela gramática que se segue, que permite escrever fórmulas lógicas simples.

$$\begin{array}{l}
 (1) \quad F \rightarrow id ( L ) \\
 (2) \quad | \quad \forall id F \\
 (3) \quad | \quad \exists id F \\
 (4) \quad L \rightarrow L , id \\
 (5) \quad | \quad id
 \end{array}$$

3.a) Construa o autômato determinístico para controlar as ações de um analisador LR(1) para a gramática dada.

3.b) Apresente a tabela de um analisador sintático LR(1) para a gramática dada. Justifique se a gramática é LR(1).

3.c) Apresente a sequência de estados, o conteúdo da pilha e as ações realizadas por um analisador sintático LR(1) da gramática dada, tomando a expressão  $\forall id \exists id id ( id )$  como entrada.

4. Representação intermédia e geração de código. Considere o programa que se segue escrito na linguagem C.

```

int fib(int n) {
    if (n == 1) return 1;
    else if (n == 2) return 1;
    else return fib(n-2) + fib(n-1);
}

int main(void) {
    return fib(4);
}
  
```

4.a) Apresente uma tradução do programa completo para código de três endereços.

4.b) Identifique os blocos básicos do código apresentado na alínea anterior e construa o grafo de fluxo da função `fib`.

4.c) Construa a árvore de ativação do programa completo, incluindo os argumentos das funções.

Convenções C3E:

<code>x = y op z</code>	<code>goto L</code>	<code>call p, n</code>	$x = y[i]$
<code>x = uop y</code>	<code>if x goto L</code>	<code>return</code>	$x[i] = y$
<code>x = y</code>	<code>ifFalse x goto L</code>	<code>x = call p, n</code>	$x = &y$
	<code>if x relop y goto L</code>	<code>return x</code>	$x = *y$
	<code>param x</code>	<code>x = param[i]</code>	$*x = y$

Onde `op` denota um operador binário, `uop` denota um operador unário e `relop` denota um operador relacional.