



UNIVERSIDADE DE SÃO PAULO - ICMC

Departamento de Ciências de Computação

SCC 605 - Teoria da Computação e Compiladores - 1º Sem /2011

PROFa: Sandra Aluisio

WIKI: <http://wiki.icmc.usp.br/index.php/SCC-605>

I – PROGRAMA

A Teoria da Computação é uma disciplina que responde quais são as capacidades e as limitações dos computadores. É dividida em 3 teorias:

Teoria das Linguagens Formais e dos Autômatos

Teoria da Computabilidade

Teoria da Complexidade

Na disciplina SCC 605, veremos as **duas primeiras**. A **primeira teoria** trata das definições e propriedades de **modelos matemáticos de computação** que têm um papel fundamental em várias áreas da Computação como o processamento de textos, compiladores, definição de linguagens de programação, dentre outras. Veremos a **Hierarquia de Chomsky** com as Linguagens Regulares, Livres de Contexto, Sensíveis ao Contexto e Irrestritas (ou com Estrutura de Frase) seus **Geradores**: Gramáticas Regulares, Livres de Contexto, Dependente de Contexto e com Estrutura de Frase e **Reconhecedores**: Autômatos Finitos, Autômatos a Pilha e Máquinas de Turing.

Utilizaremos o pacote de ferramentas gráficas JFLAP para ajudar no aprendizado de Linguagens Formais e Autômatos (<http://www.jflap.org/>).

Os conceitos acima darão subsídios para a **Teoria da Computabilidade**, pois do ponto de vista teórico, para se definir o que é ou não computável é necessário utilizar um modelo matemático que represente o que se entende por computação. Esta teoria é centralizada na Tese de Church-Turing e nas evidências dela. Church usou um sistema chamado cálculo- λ para definir **algoritmo** e Turing fez o mesmo com o uso da Máquina de Turing (MT). As duas definições foram mostradas serem equivalentes e a conexão entre a noção informal de algoritmo (solúvel efetivamente) e a definição precisa por uma MT foi chamada Tese de Church-Turing: se um **problema algorítmico** não pode ser resolvido por uma máquina de Turing, então não existe nenhuma solução computável para ele.

Nem todos os **problemas algorítmicos**, que podem ser resolvidos em princípio, podem ser resolvidos na prática: os recursos computacionais requeridos (tempo ou espaço) podem ser proibitivos. Esta observação motiva o estudo da **complexidade computacional**, mas esta teoria não será coberta neste curso.

Veremos, também, conceitos e métodos da disciplina de **Compiladores**:

Conceitos básicos: compiladores, interpretadores, montadores, filtros, pré-processadores; estrutura conceitual de um compilador; formas de organização de um compilador; processo de execução de uma linguagem de alto-nível; compilador cruzado (cross-compiler), auto-compilável (bootstrapping), auto-residente e compiler compilers (ou translator writing systems, ou compiler generators).

Análise Léxica: vantagens da separação entre as análises léxica e sintática; erros Léxicos; especificação e reconhecimento de tokens; formas de implementar os autômatos finitos que reconhecem os tokens do programa-fonte.

Análise Sintática Descendente (ASD ou TOP-DOWN)

- com Retrocesso
- sem Retrocesso (*Predictive Parsers*):
- Procedimentos Recursivos
- Dirigida por Tabela (LL(k))

Análise Semântica, Tratamento de Erros Dependentes de Contexto e Checagem de Tipos: o componente semântico de uma LP; Tarefas da Análise Semântica; Técnicas de Implementação para **Tabelas de Símbolos**; Ações Semânticas em Compiladores Dirigidos por Sintaxe e Erros da Análise Semântica; Tarefas da Checagem de Tipos; Checagem de tipos Estática versus Dinâmica; Tipo de uma variável; Compatibilidade de tipos; conversões de tipos; Operadores sobrecarregados.

Não cobriremos a última fase de um compilador: Ambientes de Execução e Geração de Código.

II – BIBLIOGRAFIA

LIVROS TEXTOS para a Teoria da Computação:

- Hopcroft, Motwani & Ullman: [Introduction to Automata Theory, Languages, and Computation](#), 2nd. Edition. Addison-Wesley, 2001. Errata do livro em: <http://www-db.stanford.edu/~ullman/ialc.html#errata>.
- Sipser, M. [Introduction to the Theory of Computation](#). PWS, 1997. (2a edição). Errata do livro em: <http://www-math.mit.edu/~sipser/itoc-errs1.2.html>
- Harel, D. [Algorithmics – The Spirit of Computing](#). Addison-Wesley Publishing Company, 1992 (2ª edição). (Existem 3 edições similares do livro na Biblioteca do ICMC: a primeira, a sua versão reduzida e a segunda edição que traz exercícios).
- Divério & Menezes. [Teoria da Computação – Máquinas Universais e Computabilidade](#). Série Livros Didáticos 5, IF UFRGS, 2ª edição, 2000, editora SagraLuzzatto.
- Menezes, P.B. [Linguagens Formais e Autômatos](#). Série Livros didáticos 3, IF UFRGS, 4ª edição, 2001, editora SagraLuzzatto. e-Book de Linguagens Formais & Autômatos. Está disponível em <http://teia.inf.ufrgs.br/library.html> a versão digital revisada da 3ª edição do Livro Linguagens Formais e Autômatos, com todas as definições, teoremas, exemplos e figuras.

LIVROS TEXTOS para Compiladores:

- Aho, A. V. et alli - [Compilers: Principles, Techniques and Tools](#). Addison-Wesley Publishing Company, 1986. (Dragãozinho em INGLÊS)
- Aho, A. V. et alli - [Compiladores - Princípios, Técnicas e Ferramentas](#), 2ª Edição, Ed. Pearson, 2007.
- Aho, A.V.; Ullman, J.D.; Sethi, R. (1995). [Compiladores: Princípios, Técnicas e Ferramentas](#). Editora LTC. (Dragãozinho em PORTUGUÊS)
- Kowaltowsky, T. - [Implementação de Linguagens de Programação](#), Guanabara Dois, 1983.
- Louden, K.C. (2004). [Compiladores: Princípios e Práticas](#). Editora Thomson Learning.
- Price, A.M.A. e Toscani, S.S. (2001). [Implementação de Linguagens de Programação: Compilador](#). Editora Sagra Luzzatto.

III - CRITÉRIO DE AVALIAÇÃO

PROVAS: Haverá 2 provas e cada uma vale de 0 a 10. **Não teremos prova substitutiva.**

1ª Prova: 29/4; 2ª Prova: 21/6

TRABALHOS PRÁTICOS: Haverá 3 Trabalhos Práticos (Projetos 1, 2, 3), todos relacionados com a linguagem **FRANKIE** para a qual construiremos o **front-end** de um compilador. Os trabalhos serão desenvolvidos por uma equipe de 4 alunos. Cada equipe receberá a tarefa de:

T1: aumentar a gramática original de FRANKIE com extensões + gerar o analisador léxico a partir do gerador de compiladores JavaCC (<https://javacc.dev.java.net/>);

T2: gerar o analisador sintático a partir do gerador de compiladores JavaCC (<https://javacc.dev.java.net/>);

T3: implementar as rotinas de checagens de tipos e análise semântica para a gramática estendida do grupo como ações no arquivo de especificação da gramática de FRANKIE

Cada trabalho vale de 0 a 10.

LISTAS DE EXERCÍCIOS. Haverá várias listas, disponíveis na Wiki do curso.

CÁLCULO DA MÉDIA:

MP = Média Aritmética das Provas

MT = Média Ponderada dos Trabalhos (T1 15%; T2 30%; T3 55%)

MF = Média Final:

Se $MP \geq 5$ então $MF = (6MP + 4MT)/10$

senão $MF = (7.5MP + 2.5MT)/10$

RECUPERAÇÃO: Só terão direito à recuperação os alunos com $3.0 \leq MF < 5.0$ e frequência $\geq$ a 70%.

Média = $MF + (Rec/2,5)$, se $Rec \geq 7,5$; ou

Média = $\text{Max} \{MF, Rec\}$, se $Rec < 5,0$; ou Média = 5,0, se $5,0 \leq Rec < 7,5$.