# MASSACHVSETTS INSTITVTE OF TECHNOLOGY

## 6.891

## Adventures in Advanced Symbolic Programming

Red Tape—Spring 2006

Issued: Wednesday, 8 February 2006

Classes: Monday, Wednesday, and Friday 1:00PM–2:00PM, Room 24-307

Leaders:
Chris Hanson, 32-383, x3-8825, cph@csail.mit.edu
Gerald Jay Sussman, 32-385, x3-5874, gjs@mit.edu
Jack Wisdom, 54-414, x3-7730, wisdom@mit.edu

Web page: http://swiss.csail.mit.edu/~gjs/6.891/
also: http://swiss.csail.mit.edu/classes/symbolic/spring06/

Readings: The readings for this subject will be taken from a variety of sources. There will be assigned readings with each problem set. All of the books we need will be on reserve in the Barton Engineering Library. Since you have had 6.001 you should have SICP (Abelson, Sussman, and Sussman; *Structure and Interpretation of Computer Programs*). Material from that book will be referred to quite often. This book is available online, but dead trees are often easier to work with. You will also need to consult the MIT/GNU Scheme documentation. This is available online.

Assignments: We will distribute an assignment on Wednesday almost every week. The default arrangement is that the assignment is due on Wednesday the next week. Most of the assignments will require the use of a computer running MIT/GNU Scheme. Our system does not run on a Mac, and we don't guarantee anything on an MS-Windows machine. We also do not trust the versions installed on Athena. However our system is tested and runs fine on GNU/Linux.

This is *free software*.[1] We will provide a site for download. For those who do not have personal computers that can run this software we will provide a way to log in to machines that have good current versions.

Projects: Just before Spring break we will assign an extended project, which will be due by the end of the term. In this project you will design and build a significant piece of symbolic-manipulation software. You will have a choice of several suggested projects. You will be expected to write elegant code that can be easily read and understood by us. You must supply a clear English explanation of how your software works, and a set of test cases illustrating and testing its operation.

Grades: The grades for this subject will be determined by a combination of classroom

[1] See http://www.fsf.org/licensing/essays/free-sw.html to find out what is meant by "free software."

participation, homework, and project work. There are no examinations in this subject. *We expect you to be at every class.*

Collaborative work: Many people learn more effectively when they study in small groups and cooperate in various other ways on homework. We are very much in favor of this kind of cooperation *so long as all participants actively involve themselves in all aspects of the work*. When you hand in a paper with your name on it we assume that you are certifying it as your work and that you were involved in all aspects of it. Even if you work with others you should do the writeup yourself, and you should indicate the names of any collaborators for each part of the assignment.

**Tentative Syllabus**

1. Embedded language philosophy
    - Ideas:
        - Generic extension of primitives
        - LAMBDA: the ultimate glue
        - Combinatory logic: Elementary Martha
    - Review of Lisp data structures and types: Numerals, Exact and Inexact, Symbols Pairs, Lists, Vectors, Hash Tables
    - PS1: XHTML combinators
2. Generic Operations
    - Dispatch:
        - Procedural style
        - Object-oriented style
        - Data-directed style
        - SOS
        - Memoization
        - Discrimination network – search
    - PS2: Generic operations on sequences

3. Rewriting systems

    - Pattern Matching
        - Matcher combinators
        - Element variables
        - Segment variables – search
    - PS3: Matcher system
    - Rule Interpretation
        - inside-out vs outside-in
        - Termination – Lyapunov function
        - Normal/Canonical forms
    - PS4: Rule system

4. Search and Backtracking

    - Directed Graphs–implicit/explicit
    - Search Patterns
        - Depth First
        - Breadth First
        - Best First
    - Loop avoidance: Theseus and Ariadne
        - hashing
        - Memoization and Dynamic Programming
    - Simple Backtracking Technology
        - SAMEFRINGE? problem
        - Generators and Consumers
        - Streams and Coroutines
        - Success and failure continuations
    - PS5: Search
    - AMB interpreter

5. Algebra

- Canonical Forms
- Intermediate Expression Bulge
- Memoization, the time-space tradeoff
- Multivariate Polynomial Arithmetic
  - Sparse and Dense Representation, using both!
  - Rational functions and GCDs
  - Interpolation: Abstracting from examples
  - GCD: Zippel algorithms
- PS?: ???
- General simplification (complication?)
  - Rational canonical forms
  - Integrating rule-based simplification
- Factoring
- Grobner basis
- PS: Projects assigned. suggestions:
  - Symbolic integration: elementary SAINT type
  - Maintaining assumptions in algebra (will see TMS later)
  - More expressive rule language for simplifier
  - Trigonometric Identities

6. Control Structures
    - Explicit Underlying Continuations
    - Coroutines and dynamic variables
    - (Pseudo) Parallel execution
        - Randomness
        - Speculative Search
    - PS?: ???
7. Deductive systems
    - Propositional Calculus
    - Rules of Inference
    - Models, Validity, Satisfiability
    - TMS
        - Explanation of Reasoning
        - Dependency-directed backtracking
    - Control of Reasoning
        - Constraint Propagation
        - Pattern-Directed Invocation
    - PS?: TMS in Algebraic Manipulation
    - Predicate Calculus
    - Matching Patterns against Patterns
        - As equations
        - Substitution-Instance Relation
        - Most General Common Specialization (Unification)
        - Most Special Common Generalization
    - Control of Reasoning
        - British Museum Algorithm
        - Pattern-Directed Invocation
        - Meta Rules
        - Amord
    - PS?: Deadbeat Dad and his Friends
    - Persistence and URIs
8. Prospects for Symbolic AI

Readings will be chosen from

1. SICP: Abelson, Sussman, and Sussman; *Structure and Interpretation of Computer Programs*
2. R5RS: Kelsey, et.al.; *Revised* $^5$ *Report on the Algorithmic Language Scheme*
3. SOS: Hanson; *Scheme Object System*
4. ART: Springer and Friedman; *Scheme and the Art of Programming*
5. RZ: Zippel; *Effective Polynomial Computation*
6. TMS: Hanson and Sussman; *Truth Maintenance and Constraints*
7. BPS: Forbus and deKleer; *Building Problem Solvers*
8. CONS: Steele; *Constraints*, MIT PhD thesis
9. LOGIC: Suppes; *Introduction to Logic*
10. AMORD: deKleer, Doyle, Rich, Steele, and Sussman; *AMORD: A Deductive Procedure System*
11. CMMR: Bundy; *The Computer Modelling of Mathematical Reasoning*

## Sign-up sheet

We need the following information from you to help us organize this subject and to allow us to install you in our computer system. Please fill out this form and hand it in at the end of the class today.

Name:

Email:

Course:

Year:

MIT address:

MIT phone:

Preferred login name:

Are you registered for 6.891? yes no

Do you need access to a computer running GNU/LINUX? yes no

Have you had 6.001 or equivalent? yes no

Have you had 6.034 or equivalent? yes no