MASSACHVSETTS INSTITVTE OF TECHNOLOGY

6.945/6.905

Adventures in Advanced Symbolic Programming

Formally: Large-scale Symbolic Systems

Red Tape—Spring 2022

Issued: Monday, 31 January 2022

Classes: Monday, Wednesday, and Friday 2:00PM, in 35-225

Leaders:
Gerald Jay Sussman, 32G-514, x3-5874, gjs@mit.edu
Manushaqe Muco, manjola@mit.edu

Web page: http://groups.csail.mit.edu/mac/users/gjs/6.945/

Readings: The readings for this subject will be taken from a variety of sources. There will be assigned readings with each problem set.

The textbook for this class is published by MIT Press:

Chris Hanson and Gerald Jay Sussman; *Software Design for Flexibility*
https://mitpress.mit.edu/books/software-design-flexibility

Any other books we need will be on reserve in the Barton Engineering Library. You may need SICP (Abelson, Sussman, and Sussman; *Structure and Interpretation of Computer Programs*). Material from that book is foundational to the material in this class. 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 Friday the next week. For special circumstances you will have to negotiate with Ms. Muco. One of the last assignments will be required only of the students registered for 6.945, the graduate version.

Software: Most of the assignments will require the use of a computer running MIT/GNU Scheme, which runs on most systems, including GNU/Linux, and Mac OS. You can get MIT/GNU Scheme from http://www.gnu.org/software/mit-scheme/. This is *free software*.[1] We will provide any other software needed.

We can help if you have trouble with the software.

Projects: Near the middle of the term you will propose an extended project, which will be due by the end of the term. In this project you will design and build a significant piece

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

of symbolic-manipulation software. If you don't come up with a great IDEA yourself, we have some ideas that you might pursue. 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. You will present a brief summary and demo in class near the end of the term.

Grades: The grades for this subject will be determined by a combination of classroom participation, homework, and project work. There are no examinations in this subject. To receive an "A" in this subject you will have to do all of the problem sets and prepare a good final project. *We expect you to be at every class and to work every problem set.*

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 separately, and you should indicate the names of any collaborators for each part of the assignment. However, we encourage you to make the final project a team effort with a joint presentation and writeup, so please try to find collaborators early in the term for this work.

Readings: Readings may be chosen from

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

Objective

Concepts and techniques for the design and implementation of systems that are evolvable—that are easy to adapt to variations in the set of problems that they are designed to address. Exploratory behavior—means for decoupling goals from strategy. Escape from the constraints of expression-based language. Working with partially-specified entities. Managing multiple viewpoints.

Strategies include language layering, generate-test separation, and propagation. Techniques to be discussed are generic operations, combinators, backtracking, dependencies, and incremental refinement.

Content

1. Additive systems
    - Ideas
        - Generic Operations
            - Additivity
            - Danger/Paranoid Programming Style
            - Translucency
        - Language Layers
            - Implicit and Explicit Features
            - emulation
            - embedding
        - Pattern-directed Invocation
            - Rule Systems
            - Term Rewriting
    - Applications
        - Algebra, Calculus, Differential Geometry
        - Peephole Optimization
    - Implementation
        - Combinators
        - Continuation Procedures
        - Simple Backtracking
        - Dependencies and Provenance
    - Mechanisms
        - Arbitrary Association
        - Interpretation
        - Compilation
        - Matching and Instantiation
        - Unification

2. Searching
    - Ideas
        - Separating Generation and Test
        - Amb and Backtracking
    - Applications
        - Puzzles
        - Games
    - Implementation
        - Control of meaning of time
        - Concurrency
        - Indeterminacy
        - Memoization
    - Mechanisms
        - Continuations
        - Dependency-directed backtracking
3. Propagation Systems
    - Ideas
        - The Problem with Expressions
        - Partial Information
        - Constraints
        - Multiple Inconsistent Worldviews
            * Alternate Sets of Assumptions
            * Local Consistency with Global Inconsistency
    - Applications
        - Modeling
        - Electrical analysis/synthesis
        - Accountability, Social Structure Models
    - Implementation
        - Cells and Propagators
        - Generic Merge
    - Mechanisms
        - Truth Maintenance Systems
        - SAT Solving Interleaved with Computation
4. Push and Pull
    - Strict and Non-strict
    - Values and Requests
    - Underlying Continuations

Sign-up sheet

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

Name:

Email:

Phone:

Course:

Year:

Are you registered for 6.945 or 6.905? yes no

Have you written a program longer than 1000 lines? yes no

Have you used assembly language? yes no

Have you used an object-oriented language (e.g. Java)? yes no

Have you used a Lisp-based language (e.g. Scheme)? yes no

Have you used a functional language (e.g. Haskell)? yes no

Have you used a logic-programming language (e.g. Prolog)? yes no

Have you had 6.009 (or old 6.001) or equivalent? yes no

Have you had 6.034 or equivalent? yes no