

CS-257L

Nonimperative Programming: Scheme!

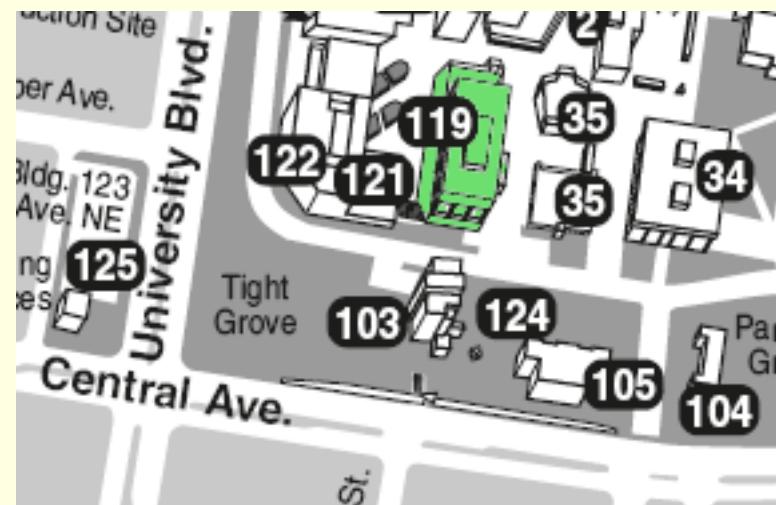
Instructor:

Joel Castellanos

e-mail: joel@unm.edu

Web: <http://cs.unm.edu/~joel/>

Office: Farris Engineering
Center (FEC) room 321



Class Web Site & WebCT

Class Web Site (Public Access)

<http://cs.unm.edu/~joel/cs150/cs257.html>

- Course Description
- Lab Assignments

WebCT (Enrolled Students Only):

<https://vista.unm.edu/webct/entryPageIns.dowebct>

- Assignment Drop-box
- Assignment discussions (blogs)
- Grades
- Lab Attendance

Textbooks

- Little Schemer - 4th Edition (Paperback)
by Daniel P. Friedman and Matthias Felleisen
Publisher: The MIT Press; 4 edition
(December 21, 1995)
- *Scheme and the Art of Programming* by
George Springer and Daniel P. Friedman
(MIT Press, 1989). You may purchase a copy
of this at the UNM Copy Center which is
located in Dane Smith Hall.

Imperative Programming

- Imperative programming is a programming paradigm that describes computation as statements that change the state of a program.
- The hardware implementation of almost all computers is imperative.
- From this low-level perspective, the program state is defined by the contents of memory, and the statements are instructions in the computer's native machine language.
- Higher-level imperative languages, such as FORTRAN and C, use variables and more complex statements, but still follow the same paradigm.
- Object-oriented languages, such as C++ and Java, add support for objects to the imperative paradigm.

A Simple State Machine - DFA

- A deterministic finite state machine or deterministic finite automaton (DFA) is a finite state machine where for each pair of state and input symbol there is one and only one transition to a next state.
- DFAs recognize the set of regular languages and no other languages.
- A DFA will take in a string of input symbols.
- For each input symbol it will then transition to a state given by following a transition function.
- When the last input symbol has been received it will either accept or reject the string depending on whether the DFA is in an accepting state or a non-accepting state.

Deterministic Finite Automaton (DFA)

A DFA is a 5-tuple, (S, Σ, T, s, A) , consisting of:

S : a finite set of states

Σ : a finite set of symbols called the alphabet

T : a transition function ($T : S \times \Sigma \rightarrow S$)

s : a start state ($s \in S$)

A : a set of accept states ($A \subseteq S$)

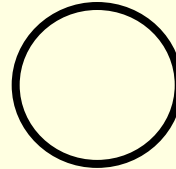
Let M be a DFA such that $M = (S, \Sigma, T, s, A)$, and $X = x_0x_1 \dots x_n$ be a string over the alphabet Σ .

M accepts the string X if a sequence of states, r_0, r_1, \dots, r_n , exists in S with the following conditions:

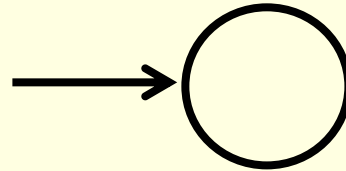
1. $r_0 = s$
2. $r_{i+1} = T(r_i, x_i)$, for $i = 0, \dots, n-1$
3. $r_n \in A$.

Directed Graph Notation for DFA

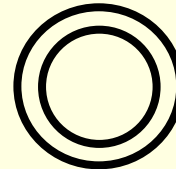
■ State



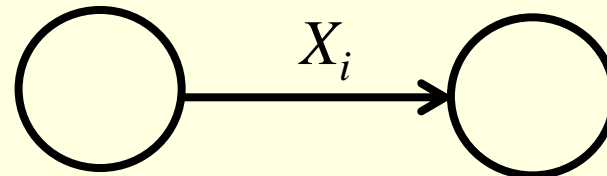
■ Start State



■ Accepting State



■ Transition



DFA Example

DFA M used the binary alphabet and recognizes strings that contain an even number of 0s.

■ $M = (S, \Sigma, T, s, A)$ where

■ $S = \{S_1, S_2\}$,

■ $\Sigma = \{0, 1\}$,

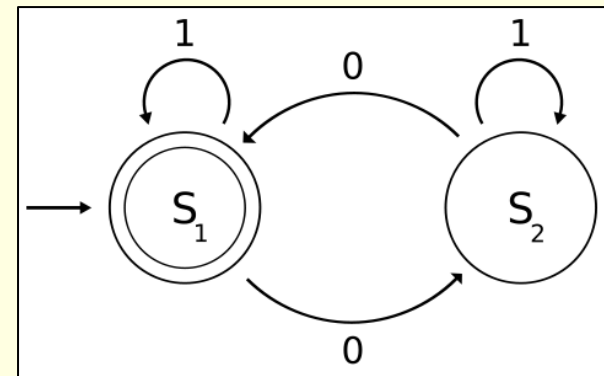
■ $s = S_1$,

■ $A = \{S_1\}$, and

■ T is defined by the following state transition table:

| | 0 | 1 |
|----------------------|----------------|----------------|
| S₁ | S ₂ | S ₁ |
| S₂ | S ₁ | S ₂ |

- S_1 represents that there has been an even number of 0s in the input so far,
- S_2 signifies an odd number.
- A 1 in the input does not change the state of the automaton.



Hierarchy of State Machines

- A DFA is very limited. It cannot, for example, recognize the language that consists of properly paired brackets, such as $((\))$.
- Non-deterministic Finite Automata (NFA) have no more power than a DFA.
- A Pushdown Automaton (PDA) is a DFA with a stack. A PDA is more powerful than a DFA and can recognize the bracket language (but cannot recognize $D = \{ww \mid w \in \{0,1\}^*\}$)
- .
- .
- .
- Turing Machine
- .

Turing Machine

- A finite alphabet that contains a special blank symbol.
- A finite number of states including a unique start state.
- A state register that stores the current state.
- A Tape
 - Divided into cells, one next to the other.
 - Each cell contains a symbol from the alphabet.
 - The tape is arbitrarily extendable to the left and to the right.
 - Cells not previously written contain the blank symbol.
- A Head
 - That can read and write symbols on the tape.
 - That can move left or right one cell at a time.
- A Table: Given a state and the symbol just read, tells the machine to:
 - (i) either erase or write a symbol, and then
 - (ii) move the head one step left or one step right, and then
 - (iii) assign a value to the state register.

Scheme

- This course uses Scheme, a multi-paradigm programming language that is best known for its support of *functional programming*.
- Functional programming is a programming paradigm that treats computation as the evaluation of mathematical functions and avoids state and mutable objects (objects that can be changed after being created).
- Functional programming emphasizes the application of functions, in contrast with the imperative programming style that emphasizes changes in state.

The Little Schemer

- Is it true that this is an atom?

atom

- Yes,

Because `atom` is a string of characters beginning with the letter `a`.

- Is it true that this is an atom?
turkey

- Yes,
because **turkey** is a string of characters
beginning with a letter.

■ Is it true that this is an atom?

1492

■ Yes,

because 1492 is a string of digits.

■ Is it true that this is an atom?

u

■ Yes,

because u is a string of one character,
which is a letter.

-
- Is it true that this is an atom?

`*abc$`

- Yes,
because `*abc$` is a string of characters beginning with a letter or special character other than a left "(" or right ")" parenthesis.

- Is it true that this is a list?
(atom)

- Yes,
because (atom) is an atom enclosed by
parentheses.

- Is it true that this is a list?
(atom turkey or)

- Yes,
because it is a collection of atoms
enclosed by parentheses.

- Is it true that this is a list?
(atom turkey) or

- No,

because these are actually two S-expressions not enclosed by parentheses. The first one is a list containing two atoms and the second one is an atom.

Beyond Chapter 1

- What is
(cdr (it rains every day))

- Nothing, because *it*, *rains*, *every*, and *day* have not been defined.

-
- What is
`(cdr '(it rains every day))`

 - `(rains every day)`, because a single quote prevents evaluation of the arguments.

-
- What is

(define p '((a b c) x y z))

(cdr p)

- (x y z)

Grading

- This course is primarily a programming course.
- There will be weekly programming assignments:
 - Fractals
 - Artificial Intelligence based games
- Programming Projects / Homework: 60%
- Quizzes: 20%
- Final Exam: 20%

Homework

- Due Sunday Night at Midnight (1/27/2008)
 - Read “The Little Schemer” Chapter 1.
 - Verify the Examples in MzScheme or DrScheme.
 - Make me a peanut butter and jelly sandwich.
 - Create 3 original questions/answers in the style of the text.
 - Use only syntax introduced in chapter 1.
 - Submit your peanut butter and jelly into WebCT.
 - Use plain text or Adobe Acrobat (pdf) format.
 - Grading:
 - C: Follow the rules.
 - B: One is Creative, Insightful, Thought Provoking.
 - A: Two or Three are C/I/TP.