

Examination 1 Programming Languages and Systems Concepts,
CSI 511
Fall 2001 (October 24, 2001)

1 Rules of the Exam

This examination is open book and notes. Calculators are permitted. Networked devices are strictly prohibited. The questions are marked as to their relative value, the exam will be scored out of 100% but is worth 25 points towards your course grade. Relax and try to do what you can.

2 The Problem Set

1. Readings (total 15 %) Note: Word for word copying from the readings will not get credit, answer IN YOUR OWN WORDS:
 - (a) Give some cases where it is better to use a scripting language than a systems programming language? Do you agree with Ousterhout?(5 %)
 - (b) Meyer's design by contract paradigm focuses on deriving preconditions, postconditions and invariants. Should preconditions be strong or weak? Should postconditions be strong or weak? Why? (5 %)?
 - (c) What does Graham think the power of Lisp compares to other languages? Why?(5 %)?
2. Theoretical Foundations (Automata) (25%)
 - (a) Consider the following languages, are they regular, why or why not (10%?)
 - i. $a^i b^i$ where $0 \leq i \leq n$ and n is finite (hint: consider a case where n is small) (5 %)
 - ii. $a^n b^n$ wher $0 < n$ (5 %)
 - (b) Sort the ordering in terms of increasing power (that is their ability to recognize languages): Nodeterministic finite state automata, deterministic push down automata, deterministic finite state automata, non deterministic pushdown automata (5 %).
 - (c) Is it possible to convert the non deterministic finite state automaton shown in Figure 1 into a deterministic finite state automaton? If so, give the corresponding deterministic finite state automaton and the regular expression for the language accepted (10 %).
3. Lisp (10 %):
 - (a) What are the results of the following operations in Lisp (write error if it would generate an error, and say why) (5 %):
 - i. (CDR (MUST GET AN A))

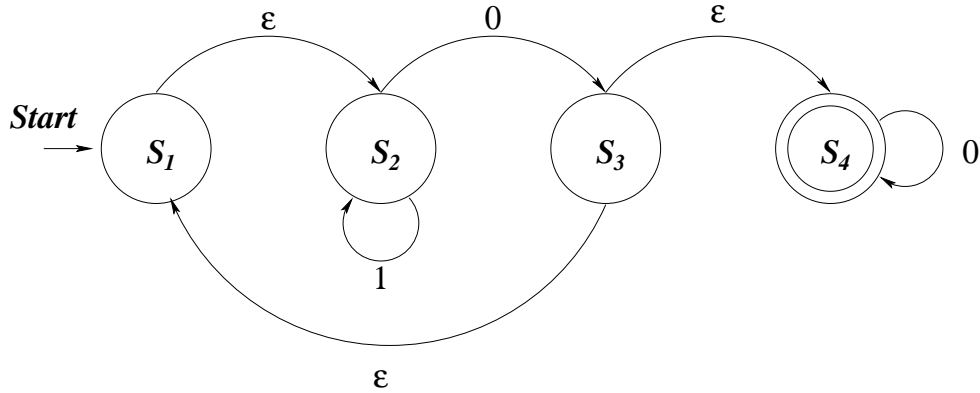


Figure 1: Non Deterministic Finite Automata for Problem 2c

ii. (CDR (QUOTE (MUST GET AN A)))

iii. (CAR (1 2 3 4))

(b) What does the QUOTE Operator do in Lisp (short answer) (5 %)?

4. Compiler Construction Tools (30 %):

(a) Give a brief description of what a lexical analyzer does (5 %)

(b) Give a brief description of what a parser does. (5 %)

(c) Consider the following simple calculator language, derive pseudocode for a Flex and Bison (Lex and Yacc) style lexical analyzer and parser (you don't have to specify Bison actions) (20 %):

```

Program ::= StatementList
StatementList ::= Statement StatementList | Statement
Statement ::= Id '=' Expr | 'read' Id | 'write' Expr
Expr ::= Term | Expr AddOp Term
Term ::= Factor | Term MultOp Factor
Factor ::= '(' Expr ')' | Id | Literal
AddOp ::= '+' | '-'
MultOp ::= '*' | '/'
Id ::= Letter | Letter IDSuffix
IDSuffix ::= Letter IDSuffix | Number IDSuffix | Letter | Number
Letter ::= [A-Za-z]
Literal ::= '-' Unsigned | Unsigned | 0
Unsigned ::= Digit | [1-9] DigitString
DigitString ::= Digit DigitString | empty
Digit ::= [0-9]
  
```

5. Type Systems And Memory Management (10 %)

(a) Consider the following code segment using C like syntax:

```

struct {
    double x;
    double y;
} PointA;

struct {
    double radians;
  
```

```

    double distance;
} PointB;

PointA.x = 1;
PointB.y = 2;
PointB.radians = 1;
PointB.distance = 2;

if (PointA == PointB){
    printf('PointA == PointB\n');
} else {
    printf('PointA != PointB\n');
}

```

What would the outcome be if a compiler using type name equivalence was handed this system? What about if structural equivalence was used (5 %)?

- (b) Consider the following program, draw the stack and show the activation records for the program after the user types in "abc" (5 %).

```

#include <stdio.h>

void p(){
    char ch = getchar(); /* read a character of input */
    if (ch != '.')
    {
        p();
        putchar(ch); /* write a character to output */
    }
}

int
main(){
    p();
    return 0;
}

```

6. Miscellaneous(10 %):

- (a) Why do most computers use twos complement instead of ones complement (2 points)?
- (b) Why do compiler writers use static scope (3 %)?
- (c) What does Binding Time mean (5 %)?