Computer Science Department
Stanford University
Comprehensive Examination in Automata, Languages, and Mathematical Theory of Computation
Autumn 1991

October 7, 1991

READ THIS FIRST!

1. You should write your answers for this part of the Comprehensive Examination in BLUE BOOKS. There are four problems in the exam; use a SEPARATE blue book for each problem. Be sure to write your MAGIC NUMBER on the cover of every blue book that you use.

2. Be sure you have all the pages of this exam. There are 3 pages.

3. The number of POINTS for each problem indicates how elaborate an answer is expected. The total number of points is 60, and the exam takes 60 minutes. This "coincidence" can help you plan your time.

4. This exam is OPEN BOOK.

5. Show your work, since PARTIAL CREDIT will be given for incomplete answers.

Comprehensive:

Automata, Languages, and Mathematical Theory of Computation (60 points) **Autumn 1991**

Problem 1 (24 points). Consider a programming language built up using assignment, composition, while, if and the local variable statement

begin new x; S end

Assume the obvious Hoare axioms

Axiom 1: Assignment Axiom

$$\{p[t/x]\}\ x := t\ \{p\}.$$

Rule 2: Composition Rule

$$\frac{\{p\}\ S_1\ \{r\}\ ,\ \{r\}\ S_2\ \{q\}}{\{p\}\ S_1\ ;\ S_2\ \{q\}}.$$

Axiom 3: *if-then-else* Rule

$$\frac{\{p \wedge e\}\ S_1\ \{q\}\ ,\ \{p \wedge \neg e\}\ S_2\ \{q\}}{\{p\}\ \textit{if } e \textit{ then } S_1 \textit{ else } S_2 \textit{ fi}\ \{q\}}.$$

Rule 4: *while* Rule

$$\frac{\{p \wedge e\}\ S\ \{p\}}{\{p\}\ \textit{while } e \textit{ do } S \textit{ od}\ \{p \wedge \neg e\}}.$$

Rule 5: Consequence Rule

$$\frac{p_0 \rightarrow p_1 \quad \{p_1\}\ S\ \{q_1\} \quad q_1 \rightarrow q_0}{\{p_0\}\ S\ \{q_0\}}$$

and

Rule 16: Variable Declaration Rule

$$\frac{\{p \wedge y = \omega\}\ S[y/x]\ \{q\}}{\{p\}\ \textit{begin new } x; S \textit{ end}\ \{q\}} \qquad \text{where } y \notin \textit{free}(p, S, q).$$

together with a background theory T (cf. Rule 5). Two programs P1 and P2 are said to provably satisfy the same triples iff for all p, q:

$$\vdash \{p\}P1\{q\} \text{ iff } \vdash \{p\}P2\{q\}.$$

We abbreviate this to $P1 \sim P2$. Justify your answers briefly in the following questions.

1a. (4 points). Clearly $\sim$ is an equivalence relation. Is it necessarily a congruence (i.e., do the programming primitives (; *while* ...) preserve this relation)?

1b. (4 points). If $P1 \sim P2$ then are they necessarily equivalent semantically (i.e., as maps from stores to stores) for all models of T?

Are the following programs $\sim$ equivalent?

1c. (4 points). x not free in P.

$P1$ = begin new x; P; end
$P2 = P$

1d. (4 points). x not free in P

$P1$ = begin new x;
$x := 0$;
P;
if $x = 0$ then diverge fi;
end
$P2$ = diverge

1e. (4 points).

$P1$ = begin new x; $P[x/y]$; $y := x$; end

$P2 = P$

where P is a program.

1f. (4 points).

$P1$ = begin new x; begin new y; $x := 0$; $y := 0$; $P[x/z_0, y/z_1]$; end

$P2$ = begin new x; begin new y; $x := 0$; $y := 0$; $P[y/z_0, x/z_1]$; end

Problem 2 (16 points). Below is text that defines the concept of ordered tree and the insert operation within the Boyer-Moore logic. The shell principle is used to introduce an abstract data type, OT. Elements of OT are either empty (ET) or constructed from a numeric label together with left and right subtrees. A predicate ORDERED.TREE is introduced to specify the elements of the OT shell that are ordered trees. Finally the INSERT function is defined and a theorem, ORDERED.TREE.INSERT is conjectured. The theorem states that INSERT applied to a number and an ordered tree returns an ordered tree.

2a. (4 points). Give at least 5 axioms assumed by the theorem prover as a consequence of the Shell Definition.

2b. (6 points). State a justification that the theorem prover might use to accept the definition of the ORDERED.TREE predicate.

2c. (6 points). Give an instance of the induction principle that could be used by the theorem prover to prove ORDERED.TREE.INSERT.

Shell Definition.

Add the shell OT of three arguments with bottom object ET, recognizer OTP, accessors LT, LABEL, and RT, type restrictions (ONE.OF OTP), (ONE.OF NUMBERP), and (ONE.OF OTP), and default values ET, ZERO, ET.

Definition.

```
(ORDERED.TREE X)
 =
(IF (NOT (OTP X))
    F
    (IF (EQUAL (ET) X)
        T
        (AND (ORDERED.TREE (LT X))
             (ORDERED.TREE (RT X))
             (OR (EQUAL (LT X) (ET)) (LESSP (LABEL (LT X)) (LABEL X)))
             (OR (EQUAL (RT X) (ET)) (LESSP (LABEL X)) (LABEL (RT X)))))))
```

Definition.

```
(INSERT L X)
 =
(IF (NOT (OTP X))
    (ET)
    (IF (EQUAL (ET) X)
        (OT (ET) L (ET))
        (IF (LESSP (LABEL X) L)
            (OT (LT X ) (LABEL X) (INSERT L (RT X)))
            (IF (LESSP L (LABEL X))
                (OT (INSERT L (LT X)) (LABEL X) (RT X ))
                X))))
```

```
Theorem ORDERED.TREE.INSERT.
    (IMPLIES (AND (NUMBERP L) (ORDERED.TREE X))
             (ORDERED.TREE (INSERT L X)))
```

Problem 3 (10 points). A minimized deterministic finite state machine accepts all strings that contain a certain fixed string s of length L. What are the largest and smallest number of states the machine might have? Justify your answer.

Problem 4 (10 points).

Consider the following program for finding a real-number approximation to the square root of a non-negataive real number r.

$z \leftarrow 0$;
$v \leftarrow max(r, 1)$;
while $\epsilon \leq v$ *do* $v \leftarrow v/2$;
    if $(z + v)^2 \leq r$ *then* $z \leftarrow z + v$;
return(z)

4a. Write a specification that expresses what this program is doing.

4b. Provide an inductive assertion, well-founded set, and partial function adequate for proving the total correctness of this program. It is not necessary to give the proof.