Fall 2017 CSE Qualifying Exam
Core Subjects

Architecture (513)

Not given in Fall 2017.

Compilers (531)

1. **LR-Parsing.** Consider the following augmented grammar G with start symbol S':

$$\begin{array}{rcl} S' & \rightarrow & S \\ S & \rightarrow & i\ S \\ S & \rightarrow & i\ S\ e\ S \\ S & \rightarrow & a \end{array}$$

   (a) Give FIRST(S) and FOLLOW(S) with respect to G.

   (b) Generate all of the LR(1) sets of items along with complete transition information for a canonical LR(1) parser for G.

   (c) Using the sets-of-items constructed in part (b), construct the action table and describe any conflicts.

   (d) Describe in detail how an arbitrary LR parsing algorithm will proceed in general when the next token is t and the stack contents are $s_0, s_1, \ldots, s_{top-1}, s_{top}$.

2. **Syntax-Directed Translation.** The do-if statement in a C-like language has the following syntax:

$$S \ \rightarrow \ \textbf{do}\ S_1\ \textbf{if}\ B$$

   At run time, it tests the Boolean expression B *first*. If this is true, then the statement S_1 is executed once; otherwise, nothing happens. Give semantic actions necessary for generation of intermediate or assembly code for this statement, assuming a one-pass compiler. As always with a one-pass compiler, the code for statement S_1, which could

be quite long, is generated while S_1 is being parsed and before B is parsed. You are allowed to use intermediate actions and place attributes on the semantic stack, as with yacc/bison.

3. **Control Flow and Liveness Analysis.** The following fragment of 3-address code was produced by a nonoptimizing compiler:

```
 1  start:  i = 1
 2  loop1:  if i > n goto part2
 3          j = 1
 4          sum = 0
 5  loop2:  if j > i goto fin2
 6          o = i * 8
 7          o = o + j
 8          s = a[o]
 9          t = j * 8
10          v = a[t]
11          y = s * v
12          if s < n goto fin1
13          sum = sum + y
14          j = j + 1
15          goto loop1
16  fin1:   j = sum
17          goto fin2
18  fin2:   i = i + 1
19          o = i * 8
20          a[o] = sum
21          goto loop2
22  part2:  no-op
```

Assume that there are no entry points into the code from outside other than at `start`.

(a) (20% credit) Decompose the code into basic blocks $B_1, B_2, \ldots$, giving a range of line numbers for each.

(b) (30% credit) Draw the control flow graph, describe any unreachable code, and coalesce any nodes if possible.

(c) (30% credit) Is the variable `o` live just before line 9? Is the variable `y` live just before line 14? Explain. Assume that `n` and `sum` are the only live variables immediately after line 22.

(d) (20% credit) Describe any simplifying transformations that can be performed on the code (i.e., transformations that preserve the semantics but reduce (i) the complexity of an instruction, (ii) the number of instructions, (iii) the number of branches, or (iv) the number of variables).

Algorithms (750)

1. Let $T(n)$ be the function defined on the positive integers by the recurrence

$$T(1) = 1\ ,$$
$$T(n) = nT(\lfloor n/2 \rfloor) + 1 \qquad \text{(for } n > 1\text{)}.$$

   Give a reasonably simple function of n that is an asymptotically tight bound on $T(n)$ assuming n is a power of 2 (in which case the floor has no effect). There are no requirements on your bound if n is not a power of 2.

2. You are given a directed acyclic graph (dag) $G = (V, E)$ with n vertices, m edges, and nonnegative integer edge weights $w(e)$ for all $e \in E$. Describe an algorithm that prints a path in G of longest weighted length. (The weighted length of a path is the sum of the weights of the edges along the path.) You may assume that G is given by the standard adjacency list representation with vertex array $V[1 \ldots n]$ and that this array has been topologically pre-sorted, that is, there is no edge from $V[i]$ to $V[j]$ if $i \geq j$. Your algorithm should run in linear time, i.e., $O(n + m)$ time. (You may attach any attributes to the entries of V that you find useful.)

3. Let G be a connected graph, and let $T_1 \neq T_2$ be two distinct spanning trees of G. Show that for every edge $e_1 \in T_1 \setminus T_2$, there exists an edge $e_2 \in T_2 \setminus T_1$ such that $(T_1 \setminus \{e_1\}) \cup \{e_2\}$ is a spanning tree of G.

Theory (551)

1. Fix an alphabet Σ. Given two strings $x, y \in \Sigma^*$ of the same length, the *interleave* of x and y is the string

$$x\#y \;:=\; x_1y_1x_2y_2\cdots x_ny_n \;,$$

where $n = |x| = |y|$ and $x = x_1 \cdots x_n$ and $y = y_1 \cdots y_n$. For two languages $L_1, L_2 \subseteq \Sigma^*$, define

$$L_1\#L_2 \;:=\; \{x\#y : x \in L_1 \wedge y \in L_2 \wedge |x| = |y|\} \;.$$

Show that if L_1 and L_2 are both regular, then $L_1\#L_2$ is regular. [Hint: From an m-state DFA for L_1 and an n-state DFA for L_2 one can build a $2mn$-state DFA for $L_1\#L_2$.]

2. Given a Turing machine M and an input string w, let $\mathit{space}(M, w)$ be the number of cells of M's tape that are scanned at least once in M's computation on input w. (It may be that $\mathit{space}(M, w) = \infty$.) A *universal space bound* is a function $s : \Sigma^* \to \Sigma^*$ such that, for any M and w such that $\mathit{space}(M, w) < \infty$, we have $\mathit{space}(M, w) \leq s(\langle M, w\rangle)$. (We encode natural numbers as strings in some reasonable way.)

   Show that no universal space bound can be computable.

3. Let RVC be the restriction of VERTEX COVER to graphs where all cycles have lengths that are multiples of 3. Show that RVC is NP-complete. [Hint: polynomially reduce VERTEX COVER to RVC.]