SPRING 1998
COMPUTER SCIENCES DEPARTMENT
UNIVERSITY OF WISCONSIN—MADISON
PH. D. QUALIFYING EXAMINATION

Programming Languages and Compilers
Qualifying Examination

Monday, February 2, 1998
3:00 – 7:00 PM
1263 Computer Sciences

GENERAL INSTRUCTIONS:

1. Answer each question in a separate book.
2. Indicate on the cover of *each* book the area of the exam, your code number, and the question answered in that book. On *one* of your books list the numbers of *all* the questions answered. *Do not write your name on any answer book.*
3. Return all answer books in the folder provided. Additional answer books are available if needed.

SPECIFIC INSTRUCTIONS:

Answer 5 of 6 questions.

POLICY ON MISPRINTS AND AMBIGUITIES:

The Exam Committee tries to proofread the exam as carefully as possible. Nevertheless, the exam sometimes contains misprints and ambiguities. If you are convinced that a problem has been stated incorrectly, mention this to the proctor. If necessary, the proctor can contact a representative of the area to resolve problems during the *first hour* of the exam. In any case, you should indicate your interpretation of the problem in your written answer. Your interpretation should be such that the problem is nontrivial.

Question 1.

This question explores certain aspects of partial evaluation. It may help to recall the following fact about partial evaluation: A partial evaluator is a program *pe* such that for all two-argument programs *p*

$[\![pe]\!]\,[p,\ s\,] = p_s$, such that for all d, $[\![p_s]\!]\,d = [\![p]\!]\,[s,\ d\,]$.

(The symbols [[and]] are used in order to clarify the distinction between when a program is executed, *e.g.*, $[\![p]\!]$, and when a program is operated on as a textual object, *e.g.*, *p*.)

Part (a)

Describe three ways in which partial evaluation can speed up the execution of a program. That is, what are three optimizations that a partial evaluator may apply so that the execution of p_s on d is faster than the execution of *p* on $[s,\ d\,]$.

Part (b)

Explain why partial evaluation might slow a program down.

Part (c)

One way of speeding up the creation of specialized programs is via the notion of a *generating extension*. A program *pgen* is a generating extension for *p* if

$[\![pgen]\!]\,s = p_s'$, such that for all d, $[\![p_s']\!]\,d = [\![p]\!]\,[s,\ d\,]$.

That is, unlike normal partial evaluation of *p*, *pgen* already has *p* "built into it" so that, when supplied with an argument *s*, it creates a program p_s', where p_s' operates just like the program p_s produced via partial evaluation. (Note that p_s' and p_s are not necessarily *identical* programs; just ones with *identical behaviors*.)

Suppose `DotProduct` computes the dot product of two vectors of length N:

```
const int N = <some constant>;

int DotProduct(int x[], int y[])  // x and y assumed to be of length N
{
   int answer = 0;

   for (int i = 0; i < N; i++) {
      answer = answer + x[i] * y[i];
   }
   return answer;
}
```

Write a procedure `DotProduct-gen` that writes a version of `DotProduct`, specialized to the value of `x[]`, to the standard output:

```
void DotProduct-gen(int x[])
{
    // MISSING -- body of DotProduct-gen
}
```

Part (d)

Compared with applying a partial evaluator to *p* and *s*, why is applying *pgen* to *s* likely to be faster?

Part (e)

Suppose that *pe* is a self-applicable partial evaluator. Let $cogen =_{df} [\![pe]\!]\,[pe,\ pe\,] = pe_{pe}$. Show that $[\![cogen]\!]\,p$ yields a program that is a generating extension for *p*.

Question 2.

For this question, assume that a program is represented using a flowgraph—a directed graph with unique entry and exit nodes such that:

- the entry node has no predecessors,
- the exit node has no successors,
- all nodes are reachable from the entry node,
- the exit node is reachable from all other nodes.

Part (a)

What does it mean for node *n* to dominate node *m* in a flowgraph? (Give an example.)

Part (b)

Why might a compiler writer be interested in computing dominators in a flowgraph?

Part (c)

Recall that a dataflow-analysis problem can be specified for a program represented using a flowgraph as follows:

1. Define a lattice of dataflow facts, including a meet operator.
2. For every node n of the flowgraph, give equations that define n_{in} (the set of facts that hold before node *n*) and n_{out} (the set of facts that hold after node *n*).

Define a dataflow-analysis problem so that the solution provides dominator information; *i.e.*, for every node *n*, n_{out} is the set of nodes that dominate *n* in the flowgraph.

Question 3.

This question concerns loop-invariant code motion. Some parts of your answer may involve standard dataflow analyses (*e.g.*, constant propagation, reaching definitions). In that case, just discuss how the results of the analysis are used; it is not necessary to discuss any of the details of the analysis itself.

Part (a)

What does it mean for an expression (*e.g.*, ID_1 *op* ID_2) to be invariant with respect to a loop?

What information would an optimizing compiler need to determine whether a given expression is invariant with respect to a given loop?

Part (b)

What issues should be considered by an optimizing compiler in determining whether to move the computation of a loop-invariant expression outside the loop (*i.e.*, what conditions should be satisfied)?

What information would the compiler need to verify those conditions?

Part (c)

Now consider a loop that contains an assignment statement

ID = <*expression*>

such that the expression is invariant with respect to the loop.

What issues should be considered by an optimizing compiler in determining whether to move the entire assignment statement outside the loop (*i.e.*, what conditions should be satisfied)?

What information would the compiler need to verify those conditions?

Question 4.

Consider the simple imperative language defined below.

program → *stmt*
stmt → *stmt* ; *stmt* | ID = *exp* | * ID = *exp* | WRITE ID
exp → INTLITERAL | ID | * ID | & ID

Note that programs in this language consist of a sequence of assignment and/or write statements, and that variables contain integers or the addresses of other variables.

By definition, a program is erroneous if it attempts to write the value of either an uninitialized variable, or a variable that contains an address.

You are to give a denotational semantics for this language (either a direct semantics or a continuation semantics) so that the meaning of an erroneous program is bottom, and the meaning of a correct program is the sequence of integer values that are written.

Be sure to say whether you are defining a direct or a continuation semantics. Define the semantic algebras you use, and the types of the meaning functions for programs, statements, and expressions. Be sure that aliasing is handled correctly; *e.g.*, the meaning of the following program is the sequence (1,2).

```
P = &Z;
Q = P;
*P = 1;
write Z;
*Q = 2;
write Z
```

Question 5.

Languages like C++ and Java extend the notion of records and structs by allowing member functions.

Part (a)

How are member functions implemented? That is, how does a compiler support unlimited occurrences of some class, each with its own function members (which can be arbitrarily large and complex)?

Part (b)

C++ differentiates between virtual and non-virtual member functions. Give a simple example that illustrates how the behavior of virtual and non-virtual member functions differ. How are virtual member functions implemented?

Part (c)

Data fields in an object are normally not virtual. How could data fields be made virtual (don't forget type correctness issues)? Why do programming languages choose not to provide virtual fields?

Question 6.

One of the most frequent criticisms of the Java programming language is that it cannot be as efficient as C or C++. One reason for this is that Java mandates that there be a runtime error for any out-of-bounds array access, and for any dereference of a null object reference (pointer).

How can traditional global program optimization techniques (data flow analysis, redundant code elimination, code movement, etc) be used to ameliorate the costs of array and object checking? Be sure to describe the information that would have to be gathered, and how that information would be used.

Note: If you are not familiar with Java, you should assume that C++ has been extended with the same mandates, and you should answer the question in the context of C++.