CS243 Second Examination

Winter 2021

March 17th, 2021

This is an open-book, open-notes, open-laptop, closed-network exam. Please do not post anything on Piazza until the solutions are put up on the class website.

You will have 2 hours to complete the exam, but the exam itself is designed to take no longer than 1 hour and 20 minutes. The examination has 6 problems worth 80 points. Please budget your time accordingly. Write your answers in the space provided on the exam. If you use additional scratch paper, please turn that in as well.

Your Name: ________________________  SUNet ID: ________________________

The following is a statement of the Stanford University Honor Code:

1. The Honor Code is an undertaking of the students, individually and collectively:
   (a) that they will not give or receive aid in examinations; that they will not give or receive unpermitted aid in class work, in the preparation of reports, or in any other work that is to be used by the instructor as the basis of grading;
   (b) that they will do their share and take an active part in seeing to it that others as well as themselves uphold the spirit and letter of the Honor Code.

2. The faculty on its part manifests its confidence in the honor of its students by refraining from proctoring examinations and from taking unusual and unreasonable precautions to prevent the forms of dishonesty mentioned above. The faculty will also avoid, as far as practicable, academic procedures that create temptations to violate the Honor Code.

3. While the faculty alone has the right and obligation to set academic requirements, the students and faculty will work together to establish optimal conditions for honorable academic work.

Signature: ________________________

<table>
<thead>
<tr>
<th>Problem</th>
<th>#1</th>
<th>#2</th>
<th>#3</th>
<th>#4</th>
<th>#5</th>
<th>#6</th>
<th>Total</th>
</tr>
</thead>
<tbody>
<tr>
<td>Score</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Max</td>
<td>5</td>
<td>5</td>
<td>20</td>
<td>20</td>
<td>20</td>
<td>10</td>
<td>80</td>
</tr>
</tbody>
</table>

1
Problem 1. Garbage Collection [5 points]

For each of the metric below, is it true or false that generational garbage collection improves over the Baker’s algorithm. Explain your answer in a sentence or two.

1. Space reclaimed False. Generational GC will clean-up younger objects more frequently, but it will eventually reclaim all dead objects.

2. Overall execution time True. As the root set already includes all pointers from earlier generations, and we expect there not to be too many references into the generation being collected (as young objects die quickly), less time will be spent finding reachable objects, reducing the overall execution time.

3. Space usage False. There is more space overhead to organize objects into generations. Also, older objects that die are cleaned up less frequently, so until a full scan is performed they can be occupying space.

4. Pause time True. By only collecting when a partition fills up, it will run more frequently and thus take less time per collection.

5. Data locality True. Objects are partitioned based on age, and as young objects typically die younger this leads to larger free regions and less fragmentation.
Problem 2. Binary Decision Diagrams [5 points]

Consider the parity function $P_n$ over $n$ binary variables $x_1, x_2, ..., x_n$. $P_n$ outputs 0 if the number of variables with value 1 are even; $P_n$ outputs 1 otherwise.

1. Construct a reduced-ordered BDD for $P_3$ using the variable ordering $x_1 \geq x_2 \geq x_3$.

![Diagram of BDD for $P_3$]

2. How many nodes (including the 0 and 1 terminal nodes) are there in the reduced-ordered BDD for $P_n$ with variable ordering $x_1 \geq x_2 \geq ... \geq x_n$?

$2n + 1$.

1 node for $x_1$, $2^{n-1}$ nodes for the other $n - 1$ nodes, and 2 terminal nodes.
**Problem 3.** Software Pipelining [20 points]

Consider the following data dependency graph. Next to each instruction are the resources used by that instruction.

1. What is the minimum initiation interval imposed by the precedence constraints?
   
   3 from top loop, 3 from the bottom loop, so 3 overall.

2. What is the minimum initiation interval imposed by resource usage?
   
   3 for each resource type, so 3 overall.
3. Run the heuristic scheduling algorithm described in class on this loop. Show both the modulo reservation table and the generated code (with nops if necessary) for a single iteration of the loop.

The minimum initiation interval is 4. Modulo reservation table:

<table>
<thead>
<tr>
<th></th>
<th>A</th>
<th>E</th>
</tr>
</thead>
<tbody>
<tr>
<td>C</td>
<td>B</td>
<td></td>
</tr>
<tr>
<td>D</td>
<td>C</td>
<td></td>
</tr>
</tbody>
</table>

Single iteration: A, nop, B, nop, nop, nop, C, D, E.

When we start at T = 3, we will end up at the following table:

<table>
<thead>
<tr>
<th></th>
<th>A</th>
<th>C</th>
</tr>
</thead>
<tbody>
<tr>
<td>C</td>
<td>B</td>
<td></td>
</tr>
</tbody>
</table>

Here we’re stuck since we can’t schedule D (there’s no slack that we can play around with), so we need to bump up our initiation interval. For T = 4, we start off with:

<table>
<thead>
<tr>
<th></th>
<th>A</th>
<th>C</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>B</td>
<td></td>
</tr>
</tbody>
</table>

Here, we can’t place D one clock after C and if we place it 2 clocks after C (in the empty row), then we can’t schedule E. So we need to backtrack and place C 4 clocks later, leading to our final answer.

4. Is the initiation interval you found optimal? Explain.

Yes.

With T = 3, you cannot schedule C right after B, since it will conflict with A (due to the delay of 2 between A and B). And if you push C two clocks later, its second resource row will conflict with B. Therefore, regardless of where you place A, you must place C three clocks after B. However, the clock right after C must be populated by A, which means you cannot schedule D and are forced to go to T = 4.
**Problem 4.** Parallelization [20 points]

Consider the following program:

```java
for (int i = 1; i < N; i++) {
    for (int j = 0; j < i; j++) {
        X[i, j] = X[i+1, j] * Y[i, j+1] + X[0, j]; (1)
    }
    Y[i, i] = X[i, i]; (2)
}
```

1. Draw the iteration space for statements (1) and (2) on the same graph. Use arrows to mark data-dependencies between iterations. Label the iterations to indicate which statement they belong to.

![Iteration space diagram](image)

2. Is there any communication-free parallelism that can be obtained using affine partitioning? If yes, what is the affine partitioning for statement (1), and for statement (2)?

There is degree-1 communication free parallelism.

First, the partitioning constraints for each statement are: Partitioning for (1): \( p = c_2 \times i + c_1 \times j + c_0 \). Partitioning for (2): \( p = c_2' \times i' + c_0' \).

Next, constrain the partitioning with data dependencies.

(a) Data dependence between \( X[i, j] \) and \( X[i + 1, j] \) in (1): (Anti-dependence)

Suppose \( i = i' + 1, j = j' \). We have

\[
    c_2 \times i + c_1 \times j + c_0 = c_2 \times (i - 1) + c_1 \times j + c_0
\]

After simplification, \( c_2 = 0; c_1, c_0 \) are unconstrained by this dependence.

(b) Data dependence between \( X[i, j] \) and \( X[0, j] \) in (1): none. From the loop bounds, \( i > 0 \), so \( (i, j) \neq (0, j) \).
(c) Data dependence between $X[i, j]$ in (1) and $X[i', i']$ in (2): none. From the loop bounds, $j < i$, so $(i, j) \neq (i', i') \forall i'$. 

(d) Data dependence between $Y[i, j+1]$ in (1) and $Y[i, i]$ in (2): (Anti-dependence)

Suppose $i = i', j + 1 = i'$.

Using the fact $c2 = 0$ from (a), We have

$$c1 \times j + c0 = c2' \times i' + c0'.$$

After simplification,

$$c1 = c2'$$
$$c0 = c2' + c0'.$$

Pick $c1 = 1$, $c2' = 1$, $c0' = 0$, $c0 = 1$.


3. Assuming $p$ is the processor ID, what is the range of $p$? What is the SPMD code for each processor? You do not need to optimize the code generated; simply wrap statements with the right conditional statements.

Solution:

Range of $p$: $1 \leq p < N$.
SPMD code for processor $p$:

```c
for (int i = 1; i < N; i++){
    for (int j = 0; j < i; j++){
        if (p == j + 1){
            X[i, j] = X[i + 1, j] * Y[i, j + 1] + X[0, j];
        }
    }
    if (p == i) {
        Y[i, i] = X[i, i];
    }
}
```
Problem 5. Pointer Analysis [20 points]

Consider the following program.

```java
public class A {
    public A link;

    public void foo(A c) {
        c.link = new A(); // h1
    }

    public void bar(A d, A e) {
        d.link = e;
    }
}

public class B extends A {
    public void foo(A f) {
        f.bar(f, f);
    }

    public void bar(A g, A i) {
        g.link = new A(); // h2
        i.link = new B(); // h3
    }
}

public class Main {
    public static void main(String[] args) {
        A a = new A(); // h4
        a.foo(a);
        a = new B(); // h5
        a.foo(a);
    }
}
```

1. For each call site on lines 15, 27, 29, specify which methods are invoked when we apply a context-sensitive and flow-sensitive pointer analysis that computes the methods called on-the-fly. Distinguish clones with their IDs as subscripts. Each call site, and each method called, must be labeled by the clone ID. How many clones of A.foo will there be? How many clones of B.foo will there be?

1 clone of A.foo; 1 clone of B.foo.

See the graph below. Due to flow-sensitivity, we can immediately tell that the call on line 27 is associated with A.foo (since the only heap object that a can point to at that
point is of type A) and the call on line 29 is associated with B.foo (since the only heap object that a can point to at that point is B). So we only need one clone for A.foo and one clone for B.foo.

\[ \begin{align*}
15_0 & \quad 27_0 & \quad 29_0 \\
B.bar_0 & \quad A.foo_0 & \quad B.foo_0
\end{align*} \]

2. Repeat question (1) above, but this time, use a **context-insensitive** and **flow-sensitive** pointer analysis. How many contexts for A.foo will there be? How many contexts for B.foo will there be?

1 context for A.foo; 1 context for B.foo.

We only have one context per function in a context-insensitive analysis. The graph will be the exact same as the one above, since flow-sensitivity ensures that we constrain the type of the object at each call-site in main to a single type.

\[ \begin{align*}
15 & \quad 27 & \quad 29 \\
B.bar & \quad A.foo & \quad B.foo
\end{align*} \]

3. Repeat question (1) again, but this time use a **context-sensitive** and **flow-insensitive** pointer analysis. How many clones of A.foo will there be? How many clones of B.foo will there be?

2 clones of A.foo; 2 clones of B.foo.

Flow-insensitivity hurts us here, since we can no longer constrain the type of a at each call site in main. Therefore, conservatively, we need to assume that each call site could invoke both A.foo and B.foo.

\[ \begin{align*}
15.0 & \quad 15.1 & \quad 27.0 & \quad 29.0 \\
A.bar.0 & \quad B.bar.0 & \quad A.bar.1 & \quad B.bar.1 & \quad A.foo.0 & \quad B.foo.0 & \quad A.foo.1 & \quad B.foo.1
\end{align*} \]
4. Perform a **context-insensitive** and **flow-insensitive** pointer analysis on the program. List the **hpts** tuples that are produced by the analysis.

\[
\begin{align*}
\text{hpts}(h4, \text{link}, h1) & \quad \text{hpts}(h5, \text{link}, h1) \\
\text{hpts}(h4, \text{link}, h5) & \quad \text{hpts}(h4, \text{link}, h4) \\
\text{hpts}(h5, \text{link}, h4) & \quad \text{hpts}(h5, \text{link}, h5) \\
\text{hpts}(h4, \text{link}, h2) & \quad \text{hpts}(h4, \text{link}, h3) \\
\text{hpts}(h5, \text{link}, h2) & \quad \text{hpts}(h5, \text{link}, h3) \\
\end{align*}
\]

Points tuples from the analysis:

\[
\begin{align*}
\text{pts}(a, h4) & \quad \text{pts}(a, h5) \\
\text{pts}(c, h4) & \quad \text{pts}(c, h5) \\
\text{hpts}(h4, \text{link}, h1) & \quad \text{hpts}(h5, \text{link}, h1) \\
\text{pts}(f, h4) & \quad \text{pts}(f, h5) \\
\text{pts}(d, h4) & \quad \text{pts}(d, h5) \\
\text{pts}(e, h5) & \quad \text{pts}(e, h4) \\
\text{hpts}(h4, \text{link}, h5) & \quad \text{hpts}(h4, \text{link}, h4) \\
\text{hpts}(h5, \text{link}, h4) & \quad \text{hpts}(h5, \text{link}, h5) \\
\text{pts}(g, h4) & \quad \text{pts}(g, h5) \\
\text{pts}(i, h4) & \quad \text{pts}(i, h5) \\
\text{hpts}(h4, \text{link}, h2) & \quad \text{hpts}(h4, \text{link}, h3) \\
\text{hpts}(h5, \text{link}, h2) & \quad \text{hpts}(h5, \text{link}, h3) \\
\end{align*}
\]
Problem 6. Path-Sensitive Analysis with SMT [10 points]

Consider the following program, where `data` is a zero-indexed array with size `N`.

```c
1 void binary_search_step (int[] data, int N, int v, int &l, int &r) {
2     if (0 <= l && l < r && r < N) {
3         int mid = (l + r) / 2;
4         if ( data[mid] <= v ) {
5             l = mid;
6         } else {
7             r = mid - 1;
8         }
9     }
10 }
```

The SSA form of the program is given below:

```c
phi0 = 0 <= l0 && l0 < r0 && r0 < N;
mid = (l0 + r0) / 2;
phi1 = data[mid] <= v;
l1 = mid;
r1 = mid - 1;
l2 = phi1 ? l1 : l0;
r2 = phi1 ? r0 : r1;
l3 = phi0 ? l0 : l2;
r3 = phi0 ? r0 : r2;
```
1. Property P1: The access to data on line 4 is always in bounds.

Write the formula for P1 using the notation of the SSA program above. (You do not need to use SMTLIB syntax when writing down the formulas. We’ll accept any format as long as it is readable.)

Solution: (not phi0) || (phi0 && 0 <= mid && mid < N)

2. As the name implies, binary_search_step is intended to be used as a step in binary search. To ensure termination, this function is expected to satisfy the following property:

Property P2: Provided that the condition on line 2 holds, upon return of binary_search_step,

(a) l3 <= r3,

(b) the range [l3, r3] contains strictly fewer elements than the range [l0, r0].

Write the formula for P2 using the notation of the SSA program above.

Solution: (not phi0) || (phi0 && l3 <= r3 && r3 - l3 < r0 - l0)

3. Suppose our SMT solver has a CHECKSAT function that accepts a given formula and returns if it can be satisfied (SAT), or not satisfied (UNSAT). How do you call CHECKSAT only once to determine whether the program satisfies both properties P1 and P2?

Let F be the formula corresponding to the SSA program. The function satisfies both properties P1 and P2 iff CHECKSAT(F and NOT(P1 and P2)) is UNSAT.

4. What is the ground truth for properties P1 and P2? For each property, answer "YES" if it is always true for any valid program input, and "NO" if otherwise. Assume there is no overflow behavior in the program. Justify your answers.

P1: YES. It is always satisfied because 0 <= 1 <= mid = (l + r) div 2 <= r < N.

P2: NO. For instance, l0 = 1, r0 = 2, N = 3, v = 5, data[1] = 2, then mid = 1, data[mid] <= v, and l3 = mid = 1, r3 = r0 = 2, and l and r don’t change their values at all.