Design and Analysis of Real-Time Systems SS 2013

Jan Reineke
Andreas Abel

Deadline: Thursday, June 13, 2013, 14:15

Assignment 6

Problem 1: Loop Bound Analysis (2+4+4 Points)

In this exercise, you should perform a *loop bound analysis* using the approach by Ermedahl et al. Consider the following program:

```
a := 7;
b := 73;
c := 0;
j := 42;
while (j >= INPUT) {
    b := b + a;
    j := j - 14;
    c := 13 * b;
    j := j + a;
}
```

1. Apply program slicing.
2. Perform a value analysis for both the interval and the congruence domain. You can assume that INPUT has a value in $[0, 10]$. In how many states can the program be at the loop header?
3. Perform an invariant analysis and derive a loop bound.

Problem 2: Evict/Fill (4+4 Points)

In this exercise we consider a cache with associativity $k = 4$ that uses the FIFO replacement policy.

1. Find a cache state that contains a specific element a, and for which the access sequence (b, c, d, e, f, g) of length $2k - 2 = 6$ does not evict this element. Can you also find such a sequence of length $2k - 1 = 7$?
2. Find a cache state such that after executing the access sequence $(a, b, c, d, e, f, g, h, i, j)$ of length $3k - 2 = 10$ the cache does not contain all of the elements g, h, i, j. Can you also find such a sequence of length $3k - 1 = 11$?

Problem 3: Cache Analysis (8+4 Points)

Consider the following program:

```
read a;
read b;
read a;
if (a>b) {
    read c;
    read d;
} else {
    read e;
    read f;
}
read x;
read a;
```

1. Perform a *May* and a *Must-Analysis* on this program, assuming an LRU-cache with associativity 4 that is empty at the start of the program. Is it possible to determine whether the last access to `a` results in a cache hit or a cache miss? Does this change if we assume that the initial cache state is unknown?
2. We now assume that the cache uses the FIFO replacement policy instead. Could an analysis determine whether the last access to `a` results in a cache hit or a cache miss if the cache is empty at the start of the program? Does this change if we assume that the initial cache state is unknown?

* Problem 4: Widening (4 Bonus Points)

In class we have seen that we can apply a fixed number of rounds of standard Kleene iteration before starting to apply a widening operator. We now assume that we apply a widening operator in every other round, i.e., we alternate between applying standard Kleene iteration and widening. Is this sufficient to ensure convergence? Justify your answer!