CSE647: Testing and Verification (Fall 2000)

Scott Stoller

Homework 5 (ver 6 Oct, 13:30). Due 13 Oct.

For every problem, you must justify your answer.

Background: We discussed the following version of Overman's Alg in lecture on 5 Oct. It computes a set of transitions that is persistent in state s.

1. Select one transition $t \in enabled(s)$. Let $T = \{thread(t)\}$, where $thread(t)$ is the thread that can perform transition t.

2. For each thread $\theta \in T$, for each transition $t \in pending(s, \theta)$, for each thread $\theta' \in \Theta \setminus T$, if $(\exists op \in used(t), op' \in used(t') : op$ can-be-dependent-with $op')$, then insert θ' in T.

3. Repeat step 2 until no more processes can be added. Return $\{enabled(s, \theta) \mid \theta \in T\}$.

op can-be-dependent-with op' holds if op and op' are operations on the same object o and $\langle op, op' \rangle \in D_o$, where D_o is a valid unconditional dependency relation for o.

An improved version of Overman's algorithm is obtained by replacing "op can-be-dependent-with op'" with "$op \triangleright_s op'$", defined below. Let $used(t, o)$ be the set of operations on object o used by transition t.

$op \triangleright_s op'$ ("op' might be the first operation to interfere with op from s") holds if op and op' are operations on the same object o and there exists a sequence $s_1 \stackrel{t_1}{\rightarrow} s_2 \stackrel{t_2}{\rightarrow} s_3 \cdots \stackrel{t_{n-1}}{\rightarrow} s_n \stackrel{t_n}{\rightarrow} s_{n+1}$ such that $s_1 = s$ and $(\forall i \in [1..n-1] : \forall op_i \in used(t_i, o) : \langle op, op_i, s_i \rangle \notin D_o^c)$ and $op' \in used(t_n, o)$ and $\langle op, op', s_n \rangle \in D_o^c$, where D_o^c is a valid conditional dependency relation for o.

Problem 1. Consider a concurrent system with exactly two threads, θ_0 and θ_1, that synchronize using only a semaphore sem. The value of sem in state s_0 is 1. Also, $enabled(s_0, \theta_0) = \{t_0\}$ and $enabled(s_0, \theta_1) = \{t_1\}$. t_0 has guard $true$ and command sem.Up(). t_1 has guard $true$ and command sem.Dn(). Make no other assumptions about the future behavior of the threads.

a. Is $\{t_0\}$ persistent in s_0?

b. Can the first version of Overman's Algorithm return $\{t_0\}$? Use the best relation D_{sem} that you can justify.

c. Can the second version of Overman's Algorithm return $\{t_0\}$? Use the best relations D_{sem}^c and $\triangleright_s$ that you can justify.

Answer: a. yes. b. no. c. yes. Regarding part a, this example was discussed during one of the lectures on VeriSoft. If you have any questions, please ask. The answers to b and c are straightforward, after you find a valid dependency relation. An unconditional dependecy relation for semaphores is: up is independent with up; all other combinations of operations are dependent. A conditional dependecy relation for semaphores is: up is independent with up; down is dependent with down if $sem = 1$; up is dependent with down if $sem = 0$.

Problem 2. Same as problem 1, except for $\{t_1\}$.
Answer: a. no. b. no. c. no. Regarding part a, this example was also discussed during one of the lectures on VeriSoft.

Problem 3. Does the "running time" of state-less selective search depend on the (unspecified) order in which transitions are selected from T? For simplicity, assume the "running time" is directly proportional to the number N_t of executed transitions (a transition that is executed multiple times is counted multiple times in N_t). Assume the state space is finite and acyclic.
Answer: Yes. Let $t; w$ denote the sequence obtained by prepending t onto sequence w. Let $PS(s)$ return a set that is persistent in s Suppose:

- $t1$ and $t2$ are in $PS(s0)$
- $t1; w1$ leads from $s0$ to $s1$, and $t2$ is in $PS(s1)$.
- $t2; w2$ leads from $s0$ to $s2$, and $t1$ is in $PS(s2)$.
- $t1$ is indep with all transitions in $t2; w2$
- $t2$ is dependent with some transition in $w1$

(Drawing a diagram might be helpful.)

Suppose we select $t1$ before $t2$ from $PS(s0)$. $t2$ will be explored from $s1$. The use of sleep sets implies that $t1$ will not be explored from $s2$.

Suppose instead that we select $t2$ before $t1$ from $PS(s0)$. $t1$ will be explored from $s2$. The dependency between $t2$ and $w1$ implies that $t2$ will be explored from $s1$.

It should not be difficult to make a concrete example of this. This argument applies to state-ful search, too.