LLSPLAT: Improving Concolic Testing by Bounded Model Checking

***Abstract*—For software testing, concolic testing reasons about data symbolically but enumerates program paths. The existing concolic technique enumerates paths sequentially, leading to poor branch coverage in limited time. In this paper, we improve concolic testing by bounded model checking (BMC). During concolic testing, we identify program regions that can be encoded by BMC on the fly so that program paths within these regions are checked simultaneously.**

We have implemented the new algorithm on top of KLEE and called the new tool LLSPLAT. We have compared LLSPLAT with KLEE using 10 programs from the Windows NT Drivers Simplified and 88 programs from the GNU Coreutils benchmark sets. With 3600 second testing time for each program, LLSPLAT provides on average 13% relative branch coverage improvement on all 10 programs in the Windows drivers set, and on average 16% relative branch coverage improvement on 80 out of 88 programs in the GNU Coreutils set.

APPENDIX

Given a control flow graph (CFG) of a function, $BB_0, BB_1, \ldots, BB_n$ is a *path* of CFG if for each $0 \leq i \leq n-1$, (BB_i, BB_{i+1}) is an edge of CFG. Given an edge (a, b) of the CFG, we call a the *source* of the edge and b the *target* of the edge. A state s of a program is a function that maps each variable x in the program to a value in the domain of x. Given two states s and s', we denote $s \xrightarrow{BB} s'$ to be an execution such that by executing the instructions of BB with the initial state s, the execution ends up with the state s'. Occasionally, if we are not interested in s or s', we omit them and write $\xrightarrow{BB} s'$ or $s \xrightarrow{BB}$.

Given a formula ψ, an *assignment* m of ψ is a function that maps each variable x in ψ to a value in the domain of x. An assignment m is a *model* of ψ, denoted by $m \models \psi$, if ψ evaluates to $true$ by m.

Given a set S of variables, a version map $\mathcal{V}$ is a *renaming* function that maps each variable $x \in S$ to a variable x_α for some $\alpha \in \mathbb{N}$. We write $\mathcal{V}(S)$ to be the set of variables $\{y \mid \exists x \in S.\ y = \mathcal{V}(x)\}$. Given a version map $\mathcal{V}$ and an assignment m to the variables in $\mathcal{V}(S)$, we denote $m_{|\mathcal{V}}$ to be an assignment to the variables in S such that for each variable $x \in S$, $m_{|\mathcal{V}}(x) = m(\mathcal{V}(x))$.

We denote $s_0 \xrightarrow{BB_0} s_1 \xrightarrow{BB_1} s_2 \ldots \xrightarrow{BB_n} s_n$ to be an execution of a program.

We first prove properties of effective dominance sets and governors. We then prove properties of our BMC algorithm.

A. Properties of Effective Dominance Sets and Governors

Lemma A.1. *Given two basic blocks m and n, if $n \in Edom(m)$, then for each path from m to n, any basic block k along the path is not polluted.*

Proof. Suppose not. Then there is a path from m to n along which there is some k that is polluted. We consider two cases. Case 1: $m = n$. Then k must be n. Thus n is polluted and is not in $Edom(m)$. Contradiction. Case 2: $m \neq n$. Then $p : m \rightarrow^* k \rightarrow^+ n$. Since k is polluted and n is reachable by k, n is polluted and is not in $Edom(m)$. Contradiction. □

Lemma A.2. *For any basic block m, $Edom(m)$ is acyclic.*

Proof. Suppose not. Then there is a basic block $n \in Edom(m)$ such that n is the source of a back edge. Hence n is polluted and is not in $Edom(m)$. Contradiction. □

Lemma A.3. *Let m be a governor. The governed region $GR(m)$ is acyclic.*

Proof. Let $BB1$ and $BB2$ be the successors of m. By Lemma A.2, $Edom(BB1)$ and $Edom(BB2)$ are acyclic. Moreover, there is not any edge $a \rightarrow b$ where $a \in Edom(BB1)$ and $b \in Edom(BB2)$. Otherwise, we can construct a path $m \rightarrow BB1 \rightarrow^* a \rightarrow b$ which bypasses $BB2$, which indicates that $BB2$ does not dominate b. Similarly, we can prove that there is not any edge $a \rightarrow b$ where $a \in Edom(BB2)$ and $b \in Edom(BB1)$. Thus, $GR(m)$ is acyclic. □

Lemma A.4. *Let m be a governor. The governed region $GR(m)$ does not have any function calls.*

Proof. Let $BB1$ and $BB2$ be the successors of m. By Lemma A.1, $Edom(BB1)$ and $Edom(BB2)$ do not have function calls. Since $GR(m) = Edom(BB1) \cup Edom(BB2)$, so does $GR(m)$. □

Lemma A.5. *A governor m dominates every basic block n in its governed region $GR(m)$.*

Proof. By definition of $GR(m)$, we know that n is either dominated by $BB1$ or $BB2$ where $BB1$ and $BB2$ are the successors of m. Without loss of generality, let us assume that $BB1$ dominates n. Since m is a governor, m dominates $BB1$. Since dominance relation is transitive, m dominates n. □

B. Properties of the BMC Generation Algorithm

Given a governor gov and a basic block $BB \in GR(gov)$, note that if g_{BB} is true, then the block formula $Blks[BB]$ encodes the program logic of BB in SSA form, which leads to Lemma A.6.

Lemma A.6. *Given a program P and a governor gov, let $\mathcal{V}, \mathcal{V}'$ be the version map before and after the SSA variable renaming for BB. The following two statements hold: (1) If an assignment m with $m(g_{BB}) = true$ is a model of $Blks[BB]$, then $m_{|\mathcal{V}} \xrightarrow{BB} m_{|\mathcal{V}'}$ is an execution of P. (2) If $s \xrightarrow{BB} s'$ is an execution of P, then there is a model m of $Blks[BB]$ such that $m(g_{BB}) = true$, $m_{|\mathcal{V}} = s$, and $m_{|\mathcal{V}'} = s'$.*

Proof. Proof by induction on the number of instructions in BB. □

Given a governor gov, we prove that, for any destination $d \in Dests(gov)$, (1) the formula $\phi \wedge g_d$ where $g_d = \bigvee_{e \in Edges[d]} e$ represents all executions from gov to d, and (2) the final version of each variable $x \in AccVars(gov)$ in ϕ always represents the value of x when an execution from gov to d reaches d.

Lemma A.7. *Given a governor gov, for any topological ordering T over $GR(gov)$ and any destination $d \in Dests(gov)$, if m is a model of $\phi \wedge g_d$, then (1) we can construct an execution ρ from the governor gov to the destination d, and (2) for each variable $x \in AccVars(gov)$, if x_α is the final version of x in ϕ, then $m(x_\alpha)$ is the value of x when the execution ρ enters the destination d.*

Proof. Since m is a model of $\phi \wedge g_d$, let the set $Taken$ be $\{BB \mid m(g_{BB}) = true\}$, i.e., the set of all basic blocks whose guard g_{BB} is set to true by m. Since g_d is $true$, we know that the guard of a predecessor of d holds, the guard of a predecessor of the predecessor of d holds, and so on. This indicates that there is a path from gov to d along which the guards g_{BB} of all basic blocks $BB \in GR(gov)$ are set to $true$ by m. Moreover, the guards g_{BB} of all basic blocks $BB \in GR(gov)$ that are not shown along the path are set to $false$ by m since the guards of two successors cannot hold at the same time. Hence we know that the set $Taken$ are the intermediate basic blocks of the path from gov to d.

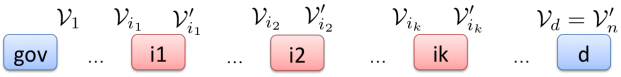

Fig. 1: Topological ordering of the governed region

As shown in Fig 1, let $BB_1, \ldots, BB_{i_1}, \ldots, BB_{i_2}, \ldots BB_n$ be the sequence of basic blocks in $GR(gov)$ sorted by the topological ordering T such that each $BB_{i_j} \in Taken$ where $1 \leq j \leq k$. Note that $gov, BB_{i_1}, BB_{i_2}, \ldots, BB_{i_k}, d$ is a path. Otherwise, T is not a topological ordering. We now construct an execution along this path. For each $BB_i \in GR(gov)$ where $1 \leq i \leq n$, let $\mathcal{V}_i$ be the version map before BB_i and $\mathcal{V}'_i$ be the one after BB_i. By Lemma A.6, we know that for each $1 \leq j \leq k$, $m_{|\mathcal{V}_{i_j}} \xrightarrow{BB_{i_j}} m_{|\mathcal{V}'_{i_j}}$ is an execution.

Fig. 2: An execution from the governor gov to a destination d

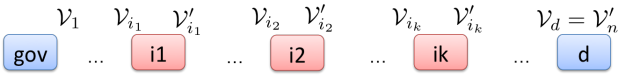

Fig. 3: Topological ordering of the governed region

Also, since the guards g_{BB} of all basic block $BB \notin Taken$ are set to $false$ by the model m, we have

$$m_{|\mathcal{V}_1} = m_{|\mathcal{V}_{i_1}}, \quad \ldots, \quad m_{|\mathcal{V}'_{i_{k-1}}} = m_{|\mathcal{V}_{i_k}}, \quad m_{|\mathcal{V}'_{i_k}} = m_{|\mathcal{V}'_n}$$

Therefore, $\xrightarrow{gov}$ $m_{|\mathcal{V}_{i_1}}$ $\xrightarrow{BB_{i_1}}$ $m_{|\mathcal{V}_{i_2}}$ $\xrightarrow{BB_{i_2}}$ $\ldots$ $\xrightarrow{BB_{i_{k-1}}}$ $m_{|\mathcal{V}_{i_k}}$ $\xrightarrow{BB_{i_k}}$ $m_{|\mathcal{V}'_n}$ $\xrightarrow{d}$ is an execution of the program P. Moreover, since $m_{|\mathcal{V}'_n} \xrightarrow{d}$, the final version of each variable x in the model m represents the value of x when the execution enters the destination d. □

Lemma A.8. *Given a governor gov, for any topological ordering T over $GR(gov)$ and any destination $d \in Dests(gov)$, if there is an execution from gov to d, then we can construct a model m for the formula $\phi \wedge g_d$.*

Proof. Let $\xrightarrow{gov}$ s_0 $\xrightarrow{BB_{i1}}$ s_1 $\xrightarrow{BB_{i2}}$ $s_2 \ldots$ $\xrightarrow{BB_{i_k}}$ s_k $\xrightarrow{d}$ be an execution from the governor gov to a destination d, as shown in Fig 2.

Let $BB_1, BB_2, \ldots, BB_n$ be the sequence of basic blocks in $GR(gov)$ sorted by the topological ordering T. Note that for $2 \leq j \leq k$, $BB_{i_{j-1}}$ must occur before BB_{i_j} along the sequence. Otherwise, T is not a topological ordering. We present this fact in Fig 3.

For each $BB_i \in GR(gov)$ where $1 \leq i \leq n$, let $\mathcal{V}_i$ be the version map before BB_i and $\mathcal{V}'_i$ be the one after BB_i. We now construct an assignment m and prove that m is a model of $\phi \wedge g_d$.

Let $Taken$ be the set $\{BB_{i_j} \mid 1 \leq j \leq k\}$. For each basic block $BB \in GR(gov)$, if $BB \in Taken$, then we set $m(g_{BB}) = true$, $m(g_{BB}) = false$ otherwise. For each variable $x \in AccVars(gov)$, we construct the assignment m in four steps:

(1) If $\mathcal{V}_1(x) = x_l$ and $\mathcal{V}_{i_1}(x) = x_h$, then for each x_α with $l \leq \alpha \leq h$, $m(x_\alpha) = s_0(x)$;
(2) For each $j \in [1, k-1]$, if $\mathcal{V}'_{i_j}(x) = x_l$ and $\mathcal{V}_{i_{j+1}}(x) = x_h$, then for each x_α with $l < \alpha \leq h$, $m(x_\alpha) = s_j(x)$,
(3) If $\mathcal{V}'_{i_k}(x) = x_l$ and $\mathcal{V}'_n(x) = x_h$, then for each x_α with $l < \alpha \leq h$, $m(x_\alpha) = s_k(x)$;
(4) For each $j \in [1, k]$, by Lemma A.6, we know that there is a model m_j of $Blks[BB_{i_j}]$ such that $m_j(g_{BB_{i_j}}) = true$, $m_{j|_{V_{i_j}}} = s_{j-1}$ and $m_{j|_{V'_{i_j}}} = s_j$. If $\mathcal{V}_{i_j}(x) = x_l$ and $\mathcal{V}'_{i_j}(x) = x_h$, then for each x_α with $l < \alpha \leq h$, $m(x_\alpha) = m_j(x)$.

Note that each variable x_α is assigned exactly once in the above construction of m, which means m does not make different values to x_α. Now we show m is indeed a model of $\phi \wedge g_d$. We consider two cases depending on whether a basic block $BB \in GR(gov)$ is in the set $Taken$.

1) Suppose $BB \in Taken$. Then BB is BB_{i_j} for some $j \in [1, k]$. First, $m \models Blks[BB_{i_j}]$ according to Step 4 of the above construction. Secondly, m evaluates $g_{BB_{i_j}}$ to true. Lastly, m evaluates $\bigvee_{c \in Edges[BB_{i_j}]} c$ to true by proving the following cases.
   a) BB_{i_j} is the left successor of the governor gov. Let $br\ e\ BB_{i_j}\ BB2$ be the terminating instruction of gov. Since

$s_0 \models e$ and $m_{|\mathcal{V}_1} = s_0$, we have $m \models \mathcal{V}_1(e)$, and thus $m \models \bigvee_{c \in Edges[BB_{i_j}]} c$.

b) BB_{i_j} is the right successor of the governor gov. This case is proved similarly as case (a).

c) BB_{i_j} is the unique successor of the basic block $BB_{i_{j-1}} \in Taken$. Since $m(g_{BB_{i_{j-1}}}) = true$ and $g_{BB_{i_{j-1}}} \in Edges[BB_{i_j}]$, we have that $m \models \bigvee_{c \in Edges[BB_{i_j}]} c$.

d) BB_{i_j} is the left successor of $BB_{i_{j-1}} \in Taken$. Let $br\ e\ BB_{i_j}\ BB2$ be the terminating instruction of $BB_{i_{j-1}}$. Since $s_{j-1} \models e$ and $m_{|\mathcal{V}'_{i_{j-1}}} = s_{j-1}$, we have $m \models \mathcal{V}'_{i_{j-1}}(e)$. Moreover, since $m(g_{BB_{i_j-1}}) = true$, then $m \models g_{BB_{i_j-1}} \wedge \mathcal{V}'_{i_{j-1}}(e)$. Hence $m \models \bigvee_{c \in Edges[BB_{i_j}]} c$.

e) BB_{i_j} is the right successor of $BB_{i_{j-1}} \in Taken$. This case is proved similarly as case (d).

2) Suppose $BB \notin Taken$. First, since $m(g_{BB}) = false$, $Blks[BB]$ are conjunctions of equations of the form $x_\alpha = x_{\alpha-1}$. By Steps (1),(2), and (3), we have $m \models Blks[BB]$. Secondly, we prove that m evaluates $\bigvee_{c \in Edges[BB]} c$ to false by contradiction. Suppose there is $c \in \bigvee_{e \in Edges[BB]} e$ such that $m \models c$. We consider the following cases depending on the form of c.

a) $c \equiv \mathcal{V}_1(e)$. Then BB is the left successor of the governor gov. Let $br\ e\ BB\ BB_{i_1}$ be the terminating instruction of gov. Since $s_0 \models \neg e$ and $m_{|\mathcal{V}_1} = s_0$, we have $m \models \neg\mathcal{V}_1(e)$, and thus $m \not\models c$. Contradiction.

b) $c \equiv \neg\mathcal{V}_1(e)$. Then BB is the right successor of the governor gov. This case is proved similarly as case (a).

c) $c \equiv g_{BB'}$. Then we know that $BB' \in Taken$ and BB is the unique successor of BB'. Since $m(g_{BB'}) = true$, then $m(g_{BB}) = true$. Contradiction.

d) $c \equiv g_{BB_u} \wedge \mathcal{V}'_u(e)$ for some $u \in [1, n]$. Since $m \models g_{BB_u}$, we know that $BB_u \in Taken$ and BB is the left successor of BB_u. Without loss of generality, let BB_u be BB_{i_j} for some $j \in [1, k]$. Then $\mathcal{V}'_u = \mathcal{V}'_{i_j}$. Let $br\ e\ BB\ BB'$ be the terminating instruction of BB_{i_j}. Note that $s_j \models \neg e$. Since $m_{|\mathcal{V}'_{i_j}} = s_j$, we have $m \models \neg\mathcal{V}'_{i_j}(e)$. Contradiction.

e) $c \equiv g_{BB_u} \wedge \neg\mathcal{V}'_u(e)$ for some $u \in [1, n]$. Since $m \models g_{BB_u}$, we know that $BB_u \in Taken$ and BB is the right successor of BB_u. This case is proved similarly as case (d).

Now that we have proved for each $BB \in GR(gov)$, $m \models g_{BB} = \bigvee_{c \in Edges[BB]} c$ and $m \models Blks[BB]$. Thus $m \models \phi$. We now prove that $m \models g_d$ where $g_d = \bigvee_{c \in Edges[d]} c$. Suppose that the destination d is the unique successor of BB_{i_k}, then $g_{BB_{i_k}} \in Edges[d]$. Since $m \models g_{BB_{i_k}}$, $m \models g_d$. Suppose that d is the left successor of BB_{i_k}, that is, the terminating instruction of BB_{i_k} is $br\ e\ d\ BB2$. Since $s_k \models e$ and $m_{|\mathcal{V}'_{i_k}} = s_k$, we have $m \models \mathcal{V}'_{i_k}(e)$. Since $g_{BB_{i_k}} \wedge \mathcal{V}'_{i_k}(e) \in Edges[d]$, we have $m \models g_d$. By the similar reasoning, if d is the right successor of BB_{i_k}, we also have $m \models g_d$. Hence $m \models \phi \wedge g_d$. □

Theorem A.9. *Given a governor gov and a destination $d \in Dests(gov)$, the formula $\phi \wedge g_d$ encodes all executions from gov to d. Moreover, the final version of each variable x in ϕ represents the value of x when an execution from gov to d enters d.*

Proof. Proved by Lemma A.7 and A.8. □