Fault Localization & Relevance Analysis

Matthias Heizmann, Christian Schilling, Numair Mansur

June 5, 2017

Definition 1 (Execution)**.** *Let π be an error trace of length n. An execution of π is a sequence of states $s_0, s_1 ... s_n$ such that $s_i, s_{i+1} \models T$, where T is the transition formula of $\pi[i]$.*
Let ϵ represent the set of all possible executions of the error trace.

Definition 2 (Blocking Execution)**.** *An execution of a trace π of size n is called a blocking execution if there exists a sequence of states $s_0, s_1 ... s_j$ where $i < j \leq n$ such that $s_i, s_{i+1} \models T[i]$, where $T[i]$ is the transition formula of $\pi[i]$ and there exits an assume statement in the trace π at position j such that $s_j \not\Rightarrow guard(\pi[j])$.*

Definition 3 (Relevancy of an assignment statement)**.** *Let β represent the set of all blocking executions of a trace π. Let there be an assignment statement of the form $x := t$ at position i. Let π' represent the trace that we get after replacing $\pi[i]$ with a havoc statement of the form $havoc(x)$ and let β' represent the set of all blocking executions for π'.*
We say that the assignment statement $\pi[i]$ is relevant if the trace after the replacement has strictly more blocked executions than the trace before the replacement, i.e if $\beta \subsetneq \beta'$.

Lemma 1. *For a program statement st and predicates P and Q, where P is condition that is true before the execution of the statement and Q is a post condition, the following two implications are equivilant(also known as the duality of WP and SP):*

$$SP(P, st) \Rightarrow Q$$
$$P \Rightarrow WP(Q, st)$$

Lemma 2. *For a predicate Q and a statement st which is an assignment statement the following implication holds:*

$$WP(\neg Q, st) = \neg WP(Q, st)$$

Lemma 3. *If a set of states have some states from which if we being the execution of a trace and that are blocking*
then

$$SP(P; trace) \not\Rightarrow guard(\pi[j])$$

Think about it more !

Theorem 1 (Relevancy). *Let π be an error trace of length n and $\pi[i]$ be an assignment statement at position i having the form $x := t$, where x is a variable and t is an expression. Let P and Q be two predicates where $P = \neg WP(False; \pi[i, n])$ and $Q = \neg WP(False; \pi[i+1, n])$. The statement $\pi[i]$ is relevant iff:*

$$P \not\Rightarrow WP(Q, havoc(x))$$

Proof. We will divide the proof in 3 cases.
Case 1: $len(\pi) = 2$***:***
Suppose we have an error trace of length $n = 2$, where $\pi[0]$ is an assignment statement of the form $x := t$ and $\pi[1]$ is an assume statement where $guard(\pi[1])$ is the guard of the assume statement. If we consider the assignment statement $\pi[0]$, P will be $\neg WP(\neg guard; x := t)$ and Q will be $guard(\pi[1])$.
"$\Rightarrow$"
If the assignment statement $\pi[0]$ is relevant, then:

$$P \not\Rightarrow WP(Q, havoc(x))$$

The relevancy of $\pi[0]$ implies that replacing $\pi[0]$ with $havoc(x)$ will result in more blocking executions then before.
i.e

$$Q \subsetneq SP(P; havoc(x))$$

Intuvitevly the above statement means that if we replace $x := t$ with $havoc(x)$ and it creates more blocking executions, then $SP(P; havoc(x))$ will have more states then in Q, while Q being $guard(\pi[1])$ here.
so,

$$SP(P; havoc(x)) \not\Rightarrow guard(\pi[1])$$

By lemma 1

$$P \not\Rightarrow WP(guard(\pi[1]); havoc(x))$$

$$P \not\Rightarrow WP(Q; havoc(x))$$

"$\Leftarrow$"
If

$$P \not\Rightarrow WP(Q, havoc(x))$$

then the assignment statement $\pi[0]$ is relevant.
Substituting the values for P and Q

$$\neg WP(\neg guard(\pi[1]), x := t) \not\Rightarrow WP(guard(\pi[1]), havoc(x))$$

From lemma 3

$$WP(guard(\pi[1]), x := t) \not\Rightarrow WP(guard(\pi[1]), havoc(x))$$

From the "duality of WP and SP" (lemma 1), if in the above expression, we consider the right hand side of the implication as P, $guard(\pi[1])$ as Q and $havoc(x)$

as st, then we can write the above implication as:

$$SP(WP(guard(\pi[1]), x := t), havoc(x)) \not\Rightarrow guard(\pi[1])$$

That means that from our supposed trace which contains one assignment statment and one assume statement with error precondition $WP(guard(\pi[1]), x := t)$, if we get a new trace where the assignment $x := t$ is replaced by $havoc(x)$, the strongest postcondition of the error precondition and $havoc(x)$ does not imply the guard of the assume statement. Which intuvitively means that replacing the assignment with havoc introduces new states which does not satisfy the guard of the assume statement. These new states also means that the number of blocking executions for the trace with *havoc* is more then the number of blocking executions in the trace with assignment. Which according to our definiton would mean that the assignment is relevant.

Case 2: $len(\pi) = n$; $\pi[i]$ is an assignment statement, where $i = 0$:
Consider the case where the length of the trace is n and the first statement $\pi[0]$ of the trace π is an assignment statement. Let $\pi_s := \pi[1, n-1]$. If we consider the assignment statement $\pi[0]$, then $P := \neg WP(false; \pi)$, $Q := \neg WP(false, \pi_s)$. Let π' be the trace where $\pi[0]$ is replaced by a havoc statement.
Let $P' := \neg WP(false; \pi') := WP(Q; havoc(x))$ and $Q' := SP(P; havoc(x))$. Observe here that P can also be stated as $WP(Q; x := t)$.
"$\Rightarrow$"
If the assignment statement $\pi[0]$ is relevant, then:

$$P \not\Rightarrow WP(Q; havoc(x))$$

i.e the trace after the replacement of $\pi[0]$ to *havoc* has strictly more blocking executions than the trace before the replacement. That means that there exists an assume statement in the trace π at position j, which is blocking more executions then before. Or we can say that there are more states s_j now for which that assume statement $\pi[j]$ is blocking.
OR

$$SP(P; \pi[0, j-1]) \subsetneq SP(P; \pi'[0, j-1])$$

$SP(P; \pi'[0, j-1])$ contains more states for which $guard(\pi'[j])$ does not hold.
Hence,

$$SP(P; \pi'[0, j-1]) \not\Rightarrow guard(\pi'[j])$$

From lemma 1:

$$P \not\Rightarrow WP(guard(\pi'[j]; \pi'[0, j-1]))$$

OR

$$P \not\Rightarrow WP(guard(\pi'[j]); \pi'[0], \pi'[1, j-1])$$

By the recursive definition of $WP()$

$$P \not\Rightarrow WP(WP(guard(\pi'[j]); \pi'[1, j-1]); \pi'[0]) \qquad (1)$$

We also know that:

$$SP(Q; \pi'[1, j-1]) \Rightarrow guard(\pi'[j])$$

and consequently from lemma (1):

$$Q \Rightarrow WP(guard(\pi'[j]); \pi'[1, j-1]) \tag{2}$$

Let $WP(guard(\pi'[j]); \pi'[1, j-1]) := R$
Then (5) and(6)can be written as:

$$P \not\Rightarrow WP(R; \pi'[0])$$

$$Q \Rightarrow R$$

From Lemma (4)

$$P \not\Rightarrow WP(Q; havoc(x)) \tag{3}$$

□