Generalized Environment: A Draft

Przemysław Daca

September 15, 2011

1 Generalized Environment

In rely-guarantee method an environmental transition can be applied any number of times, e.g. a transition that increments some variable. In LBE, where operations simply extend the path formula, such transitions could be applied infinitely many times, each time giving a distinct path formula. To cut this chain of applications the formula has to be abstracted at some point. The abstraction is at the same location as the applications, so unless covered, the transitions can be applied again. As the result, those locations become abstraction points. As test have showed, Boolean abstraction with many abstractions points exhibits low performance [1].

A promising solution is to create a single transition that overapproximates the overall effect of the environment. If subsequent application would not give more valuations, then it would be enough to apply the transition only once. LBE analysis could treat the environment as yet another transition, so there would be less abstraction locations. The precision of the environment applied to thread j and generated by thread i would be controlled by a set $\mathcal{P}_{i \triangleright j}$ of predicates.

1.1 Generating transitions

In this section, we show how transitions can be generated from abstract states. Let V be the set of all variables that occur in a program, V_g be global variables and let V_i be variables that are local to thread j. Let X' be the set X with all its elements primed. The predicates in $\mathcal{P}_{i \triangleright j}$ are expressions over $V \cup V'_g \cup V'_j$. The unprimed variables refer to the state before the environmental transition and primed elements are the values after applying the transitions. We use id_X to denote the set of equalities

$$id_X = \{v = v' \mid v \in X\}.$$

and we require that

$$\mathcal{P}_{i \triangleright j} \supseteq id_{V_g \cup V_i}.$$

We will investigate the implementation side of the problem, so let abstract states be formulas over V in static single assignment form. We also assume that variables in each thread have unique labels. Let $SP_{op}(\varphi)$ be the successor of abstract state φ by operation op. For each predicate in $\mathcal{P}_{i \triangleright j}$ we instantiate the unprimed variables in the predicate to the last SSA indexes in φ and the primed variables to the last indexes in $SP_{op}(\varphi)$. For every predicate instance i, a fresh propositional variable p_i is introduced and the equality $i \longleftrightarrow p_i$ is appended to the formula. The Boolean predicate abstraction is constructed by enumerating all models of the fresh variables in the extended formula. For a variable $v \in V_g \cup V_i$ that has not been changed

by op, the primed and unprimed versions will be the same, so the predicate $v' = v$ will follow. We implicitly associate the unprimed variables of a transition with the SSA map of φ and the primed variables with the SSA map of $SP_{op}(\varphi)$. This allows interpolation to relate the source and application formulas.

Example 1. Threads T_1 and T_2 have one global variable $V = \{g\}$ and local variables $V_1 = \{pc_1, x\}$ and $V_2 = \{pc_2, y\}$. Variables in T_1 have an upper index ^{1}, while in T_2 they have an index ^{2}. Suppose that the environmental predicates are

$$\mathcal{P}_{1\triangleright 2} = \{g' = g,\ pc_1' = pc_1,\ x' = x,\ x \geq 5,\ g' = x + 1\}.$$

Imagine that we have an abstract state in T_1

$$\varphi_1 : x^1@1 = 5 \wedge pc_1^1@1 = 2 \wedge g^1@1 < 10 \wedge y^1@1 \neq x^1@1$$

and we apply operation $op : pc_1 = 2 \xrightarrow{x:=x+1} pc_1 = 3$, thus obtaining the state

$$SP_{op}(\varphi_1) : x^1@1 = 5 \wedge pc_1^1@1 = 2 \wedge g^1@1 < 10 \wedge y^1@1 \neq x^1@1 \wedge x^1@2 = 6 \wedge pc_1^1@2 = 3. \quad (1.1)$$

The abstraction formula for $SP_{op}(\varphi_1)$ with the predicates from $\mathcal{P}_{1\triangleright 2}$ is:

$$\begin{gathered} x^1@1 = 5 \wedge pc_1^1@1 = 2 \wedge g^1@1 < 10 \wedge y^1@1 \neq x^1@1 \wedge x^1@2 = 6 \wedge pc_1^1@2 = 3 \\ \wedge p_1 \leftrightarrow g^1@1 = g^1@1 \wedge p_2 \leftrightarrow pc_1^1@2 = pc_1^1@1 \wedge p_3 \leftrightarrow x^1@2 = x^1@1 \wedge \\ p_4 \leftrightarrow x^1@1 \geq 5 \wedge p_5 \leftrightarrow g^1@2 = x^1@1 + 1 \end{gathered}$$

where $p_{1\ldots 5}$ are related to the predicates in $\mathcal{P}_{1\triangleright j}$. After abstraction the environmental transition e_1 is

$$e_1 : g' = g \wedge pc_1' \neq pc_1 \wedge x' \neq x \wedge x \geq 5.$$

The associated SSA map for the unprimed variables is

$$\{x^1 \mapsto 1, pc_1^1 \mapsto 1, g^1 \mapsto 1, y^1 \mapsto 1\}$$

and for the primed variables is

$$\{x^1 \mapsto 2, pc_1^1 \mapsto 2, g^1 \mapsto 1, y^1 \mapsto 1\}.$$

The predicate $g' = x + 1$ does not appear in e_1 since it may or may not hold.

1.2 Application and interpolation

Suppose we have a transition e from thread i and we want to apply it to abstract formula ψ in thread j. The procedure is:

1. For every variable $v \in V$ we append an equality $v^i = v^j$ to ψ, where v^j is the latest version of v in ψ and v^i is the latest version of v in the unprimed SSA map of e. By "latest version of v" we mean the one with the highest index modulo unique thread labels. These equalities will relate valuations of ψ to the unprimed variables in e.
2. Transition e is instantiated using the two associated SSA maps: the unprimed variables with the first map and the primed with the other. This gives an abstract formula that is weaker than the formula that generated the transition. Since threads have unique labels, the instantiated formula does not overlap with ψ. The instance is appended to ψ.

$SP_{op}(\varphi)$	$x^1@1 = 5 \wedge pc_1^1@1 = 2 \wedge g^1@1 < 10 \wedge y^1@1 \neq x^1@1 \wedge x^1@2 = 6 \wedge pc_1^1@2 = 3$
ψ'	$g^2@5 = 3 \wedge pc_2^2@5 = 4 \wedge y^2@5 > 0 \wedge$ $g^2@5 = g^1@1 \wedge pc_1^2@1 = pc_1^1@1 \wedge x^2@1 = x^1@1 \wedge pc_2^2@5 = pc_2^1@1 \wedge y^2@5 = y^1@1$ $g^1@1 = g^1@1 \wedge pc_1^1@2 \neq pc_1^1@1 \wedge x^1@2 \neq x^1@1 \wedge x^1@1 \geq 5 \wedge$ $g^2@6 = g^1@1 \wedge pc_1^2@2 = pc_1^1@2 \wedge x^2@2 = x^1@2 \wedge$
assert	$x^2@2 \neq 5$

Table 1: Interpolation formulas for e_1 applied to ψ and an assertion.

3. For every variable v that can be changed by thread i, that is for $v \in V_g \cup V_i$, an equality $w^i = w^j$ is append to φ, where w^i is the latest version of v in the primed SSA map of e and w^j is a new version of v in ψ. These relate the primed variables of e with the new values of variables.

Example 2. Let us apply transition e_1 from the previous example to a state ψ in thread T_2

$$\psi : g^2@5 = 3 \wedge pc_2^2@5 = 4 \wedge y^2@5 > 0. \tag{1.2}$$

In step 1. the following equalities are added to ψ:

$$g^2@5 = g^1@1 \wedge pc_1^2@1 = pc_1^1@1 \wedge x^2@1 = x^1@1 \wedge pc_2^2@5 = pc_2^1@1 \wedge y^2@5 = y^1@1. \tag{1.3}$$

Here, we assumed that variables that do not appear in SSA map, like x^2 in ψ, have a default index 1. In step 2. e_1 is instantiated to a formula $\dot{e_1}$

$$\dot{e_1} : g^1@1 = g^1@1 \wedge pc_1^1@2 \neq pc_1^1@1 \wedge x^1@2 \neq x^1@1 \wedge x^1@1 \geq 5. \tag{1.4}$$

Note that it holds that $SP_{op}(\varphi) \to \dot{e_1}$. The equalities added in step 1. relate the last valuations of ψ of to the lowest indexes of $\dot{e_1}$. In step 3. the following equalities link the latest versions of $\dot{e_1}$ with the new valuations of ψ:

$$g^2@6 = g^1@1 \wedge pc_1^2@2 = pc_1^1@2 \wedge x^2@2 = x^1@2. \tag{1.5}$$

The new formula ψ' is a conjunction of (1.2), (1.3), (1.4) and (1.5). The instantiated formula $\dot{e_1}$ in (1.4) relates the application ψ' to formula (1.1) that generated e_1, which is necessary for interpolation. Variables pc_2 and y which are local to T_2, remain unchanged by the application. Variables g, pc_1 and x receive new values specified by the constraint of e. For instance, $pc_1^2@2$ can have any value, as long it is different from $pc_1^2@1$.

Imagine that after the application of e_1 to ψ there is an assertion that $x \neq 5$ (let us skip over the detail that x is local to T_0). The new abstract state ψ' might violate the assertion, so interpolation is performed to check whether the formula trace is feasible. For the sake of simplicity let us assume that there are no other formulas involved besides $SP_{op}(\varphi)$, ψ' and the assertion. Table 1 shows the interpolation formulas for the error trace. The interpolant after $SP_{op}(\varphi)$ could be $x^1@2 > x^1@1$. Translating the SSA indexes of φ to unprimed variables and the indexes of $SP_{op}(\varphi)$ to primed variables, this interpolants can be expressed as $x' > x$. Adding this predicate to $\mathcal{P}_{1\triangleright 2}$ and restarting the analysis will make this error trace infeasible. Details of refinement depend on the interpolation procedure.

1.3 Generalizing transitions

Transitions generated in the presented way may still be applied infinitely many times. In this section we describe how the generalize multiple environmental transition into a single transition that may be applied only once.

Let $\mathcal{B}(\mathcal{P}_{i\triangleright j})$ be the set all Boolean combinations of predicates from $\mathcal{P}_{i\triangleright j}$ that have different valuations, so combinations like $p \vee r$, $r \vee p$ and $p \vee p \vee r$ are considered to be equal. We define $L_{i\triangleright j}$ as the complete lattice over $\mathcal{B}(\mathcal{P}_{i\triangleright j})$, partially ordered by $\rightarrow$, the least upper bound being $\vee$ and with formula $false$ as the bottom element. Lattice $L_{i\triangleright j}$ orders transitions by how much their application can change, for example the transition $x' = x \wedge x' > 0$ can change less than $x' = x$. The least upper bound of transitions gives a transitions that can change more than any of them. We distinguish the element e_{id} which is a Boolean abstraction of $\bigwedge id_{V_g \cup V_i}$ with the predicates $\mathcal{P}_{i\triangleright j}$. Transition e_{id} expresses that environment does not change anything.

Example 3. Let g be the only variable and lattice $L_{1\triangleright 2}$ be based on the following predicate set

$$\mathcal{P}_{1\triangleright 2} = \{g' = g,\, g > 0,\, g' = g + 1\}.$$

The identity transition is

$$e_{id} : g' = g \wedge g' \neq g + 1.$$

Now, consider transitions e_2 and e_3

$$e_2 : g > 0 \wedge g' \neq g \wedge g' = g + 1$$

$$e_3 : g \leq 0 \wedge g' \neq g.$$

Transition e_2 increments g when it is positive, while e_3 gives some new value to g provided that it was is non-positive. Transition $e_2 \vee e_3 \vee e_{id}$ has the same effect as applying e_2 or e_3 or not applying any transition.

Transitions can be applied one after another. Let $\varphi[X \rightarrow \overline{X}]$ denote an abstract state, where every occurrence of $v \in X$ has been replaced by a fresh variable $\overline{v}$. Moreover, let $\alpha_X(\varphi)$ be the Boolean abstraction of φ with the predicates from X.

Composition of transitions e' and e, denoted by $e' \circ e$, is defined as

$$e' \circ e \;=\; \alpha_{\mathcal{P}_{i\triangleright j}}(e[V' \rightarrow \overline{V}] \wedge e'[V \rightarrow \overline{V}])$$

where the set of predicates $\mathcal{P}_{i\triangleright j}$ should be clear from the context. Composition $e' \circ e$ describes the relation between the unprimed variables of e and primed variables of e' linked by an intermediate variables in $\overline{V}$. Composition of two transitions from $L_{i\triangleright j}$ also belongs to the lattice. We assume that $\circ$ binds stronger than $\vee$.

Suppose that we have generated a set of transitions E and we want to create a single transition that overapproximates all of them. This generalized transition should describe the effect of applying the transitions in an arbitrary sequence or lack of any application. We define a function $f_E : L_{i\triangleright j} \rightarrow L_{i\triangleright j}$ on transitions

$$f_E(e) \;=\; (\bigvee E) \circ e \vee e_{id}.$$

Given some transition e, f_E will return the composition of $\bigvee E$ with e, but not smaller then the identity transition e_{id}. Function $f_E(e)(\varphi)$ may be read as "apply e to φ and then apply any transition from E or do not apply anything." For a fixed E the function is monotone.

Generalized transition e_g is the least fixed point of f_E. Since lattice $L_{i\triangleright j}$ has a finite size, therefore e_g can be obtained be iterating over $f_E^n(false)$ until it inevitably stabilizes.

Example 4. Let us continue with the previous example, but assume that e_2 is the only transition, i.e. $E = \{e_1\}$. Function $f_E(e)$ becomes

$$f_E(e) \;=\; e_2 \circ e \;\vee\; e_{id}.$$

Let us iterate over f_E until the least fixed point is reached. The first iteration gives the identity transition:

$$f_E^1(false) \;=\; (e_2 \circ false \;\vee\; e_{id} = e_{id}.$$

The second iteration is

$$f_E^2(false) \;=\; f_E(e_{id}) \;=\; e_2 \circ e_{id} \;\vee\; e_{id} =$$

$$\alpha_{\mathcal{P}_{i \triangleright j}}(\overline{g} = g \;\wedge\; \overline{g} \neq g+1 \wedge \overline{g} > 0 \;\wedge\; g' \neq \overline{g} \;\wedge\; g' = \overline{g}+1) \;\vee\; e_{id} =$$
$$(g > 0 \;\wedge g' \neq g \;\wedge\; g' = g+1) \;\vee\; e_{id}.$$

The result is weaker than e_{id}, so we continue with $f_E^3(false)$:

$$f_E^3(false) \;=\; f_E(f_E^2(false)) \;=\; e_2 \circ f_E^2(false) \;\vee\; e_{id} =$$

$$\alpha_{\mathcal{P}_{i \triangleright j}}(((g > 0 \wedge \overline{g} \neq g \;\wedge\; \overline{g} = g+1) \vee (\overline{g} = g \;\wedge\; \overline{g} \neq g+1)) \wedge \overline{g} > 0 \;\wedge\; g' \neq \overline{g} \;\wedge\; g' = \overline{g}+1) \;\vee\; e_{id} =$$
$$(g > 0 \;\wedge\; g' \neq g) \;\vee\; e_{id}.$$

The fourth iteration is:

$$f_E^4(false) \;=\; f_E(f_E^3(false)) \;=$$
$$\alpha_{\mathcal{P}_{i \triangleright j}}(((g > 0 \;\wedge\; \overline{g} \neq g) \vee (\overline{g} = g \;\wedge\; \overline{g} \neq g+1)) \wedge \overline{g} > 0 \;\wedge\; g' \neq \overline{g} \;\wedge\; g' = \overline{g}+1) \;\vee\; e_{id} =$$
$$g > 0 \;\vee\; e_{id}.$$

The last iteration gives

$$f_E^5(false) \;=\; f_E(f_E^4(false)) \;=$$
$$\alpha_{\mathcal{P}_{i \triangleright j}}((g > 0 \;\vee\; (\overline{g} = g \;\wedge\; \overline{g} \neq g+1)) \wedge \overline{g} > 0 \;\wedge\; g' \neq \overline{g} \;\wedge\; g' = \overline{g}+1) \;\vee\; e_{id} =$$
$$g > 0 \;\vee\; e_{id}.$$

The result has stabilized, therefore the generalized transition e_g is

$$g > 0 \;\vee\; (g \;\wedge\; g' \neq g+1).$$

Notice that $f_E^n(false)$ overapproximates the effect of applying transitions from E less than n times in any order. The intuitive meaning of e_g is that it does not change anything or, if g is non-negative, it may set g to any value. In LBE analysis, it is enough to apply e_g once, since subsequent application will give the same result.

References

[1] D. Beyer, M. E. Keremoglu, and P. Wendler, "Predicate abstraction with adjustable-block encoding," in *Proceedings of the 2010 Conference on Formal Methods in Computer-Aided Design*, FMCAD '10, (Austin, TX), pp. 189–198, FMCAD Inc, 2010.