Limitation-I-Optimize Flowsto

Monday, May 23, 2022 2:18 PM

⇒ Redue the "flowsto" Over-approximate

I: eliminate Useless Data-Control Flow.

① Redundant If Branch.

case 1: Normalized Boolean Exp in Guard is constant.

⇒ Remove another Block.

⇒ Remove the data-control "flowsto" of variables assigned inside the Block on Variables in guard.

$x \leftarrow 0$.

if $(x<3, [y=1], [y=2])$

$\rightsquigarrow$ $\exists v, \forall \tau, \langle \tau, b \rangle \Downarrow v$

case 2: 2 Blocks in two Branches are syntactically identity.

⇒ Remove another Block.

⇒ Remove the data-control "flowsto" from all Variables in guard to variables assigned inside the Block.

if $(b, [y=1], [y=1])$.

$\rightsquigarrow$ 2 branches are the same

② Redundant While loop.

Case 1: Normalized Boolean Exp in Guard is constant false.

⇒ Remove the Block of the while Body

⇒ Remove the data-control "flowsto" from all Variables in guard to variables assigned inside the Block.

$x \leftarrow 0$

while $x > 0$,

$y = 1$.

$\rightsquigarrow$ same as if-case 1. $\exists v. \forall \tau, \langle \tau, b \rangle \Downarrow v$.

③ Redundant variables in Guard.

For b of guard in every if and while command.

$\forall x \in VAR(b). \quad x \notin VAR(NF(b))$.

Remove all the data-control "flowsto" from x to variables in the Body Blocks

$x \leftarrow q(0)$

$\rightsquigarrow$ if $([x-x=0], [y=1], [y=2])$.

variable disappeared from VAR(b), or VAR(e) or VAR(α) by normalization

II. Remove redundant data-value "flowsto".

for the e and ψ in every assignment command. $y \leftarrow e$. and $y \leftarrow \psi$

$\forall x \in VAR(e)$, $x \in VAR(NF(e))$, or $x \in VAR(\psi) \wedge x \notin VAR(NF(\psi))$

Remove the data-value "flowsto" from x to y

$x \leftarrow q(0)$

$y \leftarrow q(x-x)$.

$\rightsquigarrow$ same as boolean expression.

remove dependency after normalization.

III: Non-trival "liveness Analysis" $\rightsquigarrow$ variable Dead.

example 1

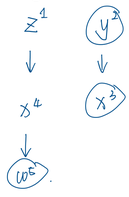
$z \leftarrow 0$

$y \leftarrow q(0)$;

$x \leftarrow q(y+1)$;

$x \leftarrow z+1$;

$w \leftarrow q(x)$.



⇒ Expected Adapt = 2

if liveness refined ⇒

⇒ Analysed Adapt = 1.

un-sound. ×

$\rightsquigarrow$ cannot further improve.

⇒ Cannot further Optimize.

examples:

Event Reexitting occurs in a longer adaptivity chain, and removing the other vertex is safe, keeping both flow doesn't lead to over-approximation.

V.

eg: $z \leftarrow q(0)$

$y \leftarrow 0$

$x \leftarrow y+1$

$x \leftarrow q(z+1)$

⇒ Expected Adapt = 2

⇒ liveness refined ⇒

⇒ Adapt = 2.

No over-approximation

IV: Non-trival Over-Approximation:

path Sensitive syntactically but not semantically:

⇒ 2 if command have the same guard, the take the same branch always.

the adaptivity Expected by $\max\{ Adap_{true\,branch\,1} + Adap_{true\,branch\,2},\ Adap_{f\text{-}branch\,1} + Adap_{f\text{-}branch\,2} \}$

the $Adapt_{program} = \max\{ Adap_{true\,branch\,1} + Adap_{true\,branch\,2},\ Adap_{f\text{-}branch\,1} + Adap_{f\text{-}branch\,2},\ A_{t\text{-}branch\,1} + A_{f\text{-}branch\,2},\ A_{f\text{-}branch\,1} + A_{t\text{-}branch\,2} \}$

eg: input a: $y \leftarrow q(0)$

if $a > 0$

$y \leftarrow q^{100}(y)$

else

$y \leftarrow q(y)$.

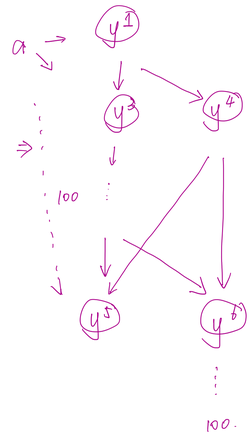
if

$a > 0$:

$y \leftarrow q(y)$

else

$y \leftarrow q^{100}(y)$.



longest path: 201. if $100 = n$ ⇒ $2n+1$

Expected Adapt: 102. ⇒ $n+2$.