The Synthesis of Loop Invariants

Program verification involves 3 tasks

a) Specification
   1) for each function that constitutes the interface
   2) for each helper function
   3) for each loop

- specification is hard
- 2 and 3 are considered onerous requirem. while 1 is generally thought to be good practice
- in some cases the specification is available implicitly
  - no integer computations should overflow
  - no array access should be out-of-bounds
  - no uninitialized var. are used
  - no NIL pointer is dereferenced
  - no dangling pointers are created
  - no garbage memory cells are created

Task b) verification condition generation

Task c) theorem proving

b) is easy if all loops are annotated with invariants

c) we have looked at.

Question today: how to create loop invariants automatically?

- can only hope to have some heuristics

Strategy A
- use only sound heuristics
  all invariants discovered are indeed invariants

Strategy B
- use potentially unsound heuristics as well
- use VCGen + ThProver to check the candidate invariants

Problem: ThProver is incomplete so we don't know if the invariant is not good or the prover is unable to prove it.
- nevertheless this can lead to some good invariants

One formulation of the invariant issue.

- a cutpoint is a point in the program
- there is a cutpoint right before each return
- there is at least a cutpoint on each circular path through the program
- we associate assertions I_k with cutpoint P_k

~~these assertions~~

Conventions

- let $\bar{x}$ be the set of variables
- for a path α we define
  - $r_\alpha(\bar{x})$ a function that gives the values of variables at end of path as a function of values at beginning
  - $p_\alpha(\bar{x})$ a predicate that says whether the path α is taken given some values of variables before the path

A set of invariants is correct if

a) for each path α from START to cutpoint k (without crossing other cutpoints)

$$\forall \bar{x}.\ \varphi(\bar{x}) \wedge p_\alpha(\bar{x}) \Rightarrow I_k(r_\alpha(\bar{x}))$$

$\varphi(\bar{x})$ is the precondition

and

b) for each path α from cutpoint i to j we have

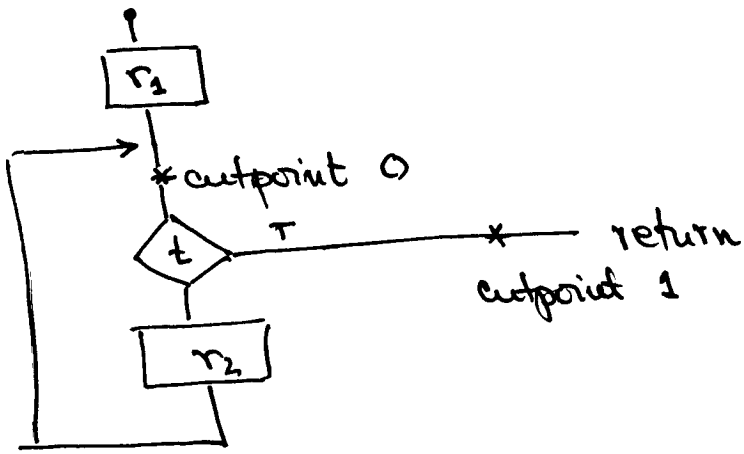
$$\forall \bar{x} \;\; I_i(\bar{x}) \wedge R_\alpha(\bar{x}) \Rightarrow I_j(r(\bar{x}))$$

---

a) says that invariants are established initially

b) says that they are maintained.

A simple example



Need

1. $\forall \bar{x}.\ \varphi(\bar{x}) \Rightarrow I_0(r_1(\bar{x}))$
2. $\forall \bar{x}.\ I_0(\bar{x}) \wedge \neg t(\bar{x}) \Rightarrow I_0(r_2(\bar{x}))$
3. $\forall \bar{x}.\ I_0(\bar{x}) \wedge t(\bar{x}) \Rightarrow I_1(\bar{x})$

These are equations that we have to solve for I_0 and I_1

I_1 might be given (the postcondition)

~~1. gives a condition on~~

Pick a candidate for I_0

- if 1 is not satisfied then I_0 is too strong
- if 3 is not satisfied then I_0 is too weak
  - it might be an invariant, but is not a useful one because is not strong enough to allow us to prove the postcondition
- if 2 is not satisfied then I_0 is just not good

Heuristics

- Start from 3 and set

$$I_0(\bar{x}) = t(\bar{x}) \Rightarrow I_1(\bar{x})$$

this is the weakest solution that satisfies 3

- Use heuristics to strengthen I_0
  - we do not need to check 3 anymore
  - we can use 1 as a sanity check
  - we use 2 to drive the heuristics

Define a family of predicates

$$P^0_{(\bar{x})} = t_1(\bar{x}) \Rightarrow I_1(\bar{x})$$

$$P^k_{(\bar{x})} = P^0_{(\bar{x})} \wedge \neg t_1(\bar{x}) \Rightarrow P^{k-1}(r_2(\bar{x}))$$

One can prove that

$P^k_{(\bar{x})}$ is the weakest predicate on $\bar{x}$ that ensures that <u>if</u> the execution starts at cutpoint 0 with $\bar{x}$ <u>and if</u> cutpoint 1 is reached after <u>at most</u> k iterations then $I_1(\bar{x}')$ is satisfied.

We can show that P^k is a chain

$$P^0 \Leftarrow P^1 \Leftarrow P^2 \cdots$$

<u>Proof</u> by induction on k.

$$P^1 \Rightarrow P^0 \quad \checkmark$$

since $P^{k-1} \Rightarrow P^{k-2}$ $\quad k \geq 2$

$$\left(\neg t(x) \Rightarrow P^{k-1}(\bar{x})\right) \Rightarrow \left(\neg t(x) \Rightarrow P^{k-2}(\bar{x})\right)$$

and then

$$P^k(\bar{x}) \Rightarrow P^{k-1}(\bar{x}) \qquad k \geq 2$$

The weakest "useful" invariant is

$$\forall k.\ k \geq 0 \Rightarrow P^k(\bar{x})$$

(but this is a strange form of quantification since $P^k(\bar{x})$ is a family of predicates more appropriately is

$$\bigwedge_{k \geq 0} P^k(\bar{x})$$

This ∞ conjunction is impossible to calculate in general.

[Note: all this material was basically weakest preconditions seen again]

What if we can find a k such that

$$\forall \bar{x} \; P^k(\bar{x}) \Rightarrow P^{k+1}(\bar{x})$$

then $$\bigwedge_{k \geq 0} P^k(\bar{x}) = P^k(\bar{x})$$

yes, since $P^k(\bar{x}) \Leftrightarrow P^{k+1}(\bar{x}) \Leftrightarrow \ldots$

This method is called induction iteration

Norihisa Suzuki, Kiyoshi Ishihata
"Implementation of an Array Bound Checker", 4th POPL (1977)

A slight improvement

- the predicates P^k becomes more complex
- it becomes more and more complicated to verify that $P^k \Rightarrow P^{k+1}$
- so we simplify P^k before using it to compute the P^{k+1}
- we can also <u>strengthen</u> P_k

$$R^0(\bar{x}) = t(\bar{x}) \Rightarrow I_1(\bar{x})$$

$$R^k(\bar{x}) = R^0(\bar{x}) \wedge \neg t(\bar{x}) \Rightarrow \text{strengthen}\left(R^{k-1}(x_2(\bar{x}))\right)$$

if $\text{strengthen}(R) \Rightarrow R$ for all R

then we still have

$$R^k(\bar{x}) \Rightarrow P^k(\bar{x})$$

- we are loosing the "weakest" property
- and the completeness

Typical strengthening heuristics

- drop disjuncts

$$\text{str}(t(\bar{x}) \Rightarrow I_1(x)) = I_1(\bar{x})$$

Integer heuristics

- try to eliminate from P^k variables that are modified in the loop (by $r_2(\bar{x})$)
- as an extreme case if P^k refers only to variables that are not modified by $r_2(\bar{x})$

then $$P^k(r_2(\bar{x})) = P^k$$

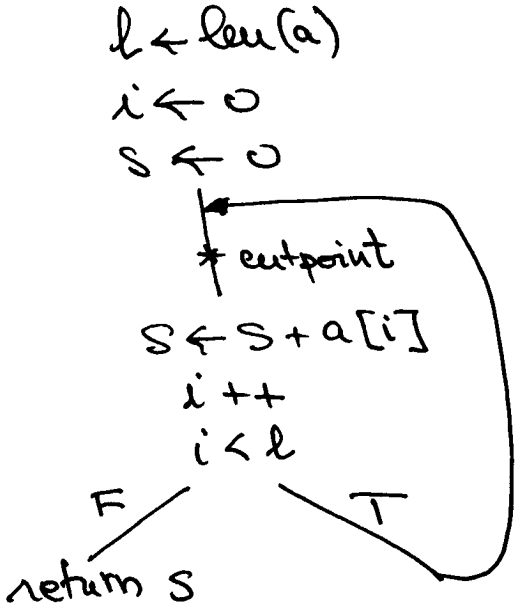
also $$P^k \Rightarrow (\neg t(x) \Rightarrow P^k)$$

$$P^k \Rightarrow P^0$$

thus $$P^k \Rightarrow P^{k+1}$$ bingo!

- one way to eliminate vars is using Fourier-Motzkin elimination if the predicates are linear inequalities and/or equalities

Example Consider the program



- take the implicit assertion that at cutpoint the invariant must imply
  - $i \geq 0$
  - $i < len(a)$
- So we start with

$P^0 = i < len(a)$ → not a tautology

$P' = P^0 \wedge (i+1 < l \Rightarrow i+1 < len(a))$

check $P^0 \Rightarrow (P^0 \wedge (i+1 < l \Rightarrow i+1 < len(a))$

need to show

$i < len(a) \wedge i+1 < l \Rightarrow i+1 < len(a)$

cannot show this

we can strengthen

$$i+1<l \Rightarrow i+1 < len(a)$$

- negate and ~~convert to ∈~~ try to show unsat.

$$i+1<l \wedge i+1 \geq len(a)$$

- use Fourier-Motzkin elimination to eliminate i (the variable that changes)

$$l > len(a)$$

- negate again $\quad l \leq len(a)$

we have $l \leq len(a) \Rightarrow (i+1<l \Rightarrow i+1 < len(a))$

$l \leq len(a)$ is stronger

$$P^2 = P^0 \wedge (i+1<l \Rightarrow l \leq len(a))$$

Now we can show that

$$P^1 \Rightarrow P^2$$, thus P^1 is an invariant and is useful

We now check condition 1. by backpropagating

$P^1 = i < len(a) \wedge l \leq len(a)$ to the START

check $0 < len(a) \wedge len(a) \leq len(a)$

Need precondition $\boxed{0 < len(a)}$