Verification Condition Generation

Testing, Quality Assurance, and Maintenance
Fall 2023

Prof. Arie Gurfinkel

based on slides by Prof. Ruzica Piskac and others

UNIVERSITY OF WATERLOO

Verification Condition Generation in a Nutshell

loop free

GCGen

SymEx or WP

Program
{A} **P** {B}

Guarded
Command
{A} $\mathbf{P_{GC}}$ {B}

Verification
Condition
$\mathbf{P_{VC}}$

FOL

$\mathbf{P_{VC}}$ is valid if and only if $\vdash$ {A} **P** {B}

Loop-Free Guarded Commands

Introduce loop-free guarded commands as an intermediate representation of the verification condition

```
c ::=  assume b
      | assert b
      | havoc x
      | c1 ; c2
      | c1 || c2
```

$c ::= \texttt{assume } b \mid \texttt{assert } b \mid \texttt{havoc } x \mid c_1 \texttt{ ; } c_2 \mid c_1 \texttt{ || } c_2$ *(non-deterministic choice)*

From Programs to Guarded Commands

GC(`skip`) =

assume true

GC(x := e) =

assume $tmp = x$; havoc x; assume ($x = e[tmp/x]$)

GC(c_1 ; c_2) =

GC(c_1) ; GC(c_2)

where tmp is fresh

GC(`if` b `then` c_1 `else` c_2) =

(assume b; GC(c_1)) || (assume :b; GC(c_2))

GC(`while` b inv I do c) = ?

Guarded Commands for Loops

GC(**`while`** *b* **inv** `I` **do** *c*) =

> assert `I`;
> havoc x_1; ...; havoc x_n;
> assume `I`;
> ((assume *b*; GC(*c*); assert `I`; assume false) ||
> assume :*b*)

where x_1, ..., x_n are the variables modified in *c*

Example: VC Generation

{n ¸ 0}

$p := 0;$

$x := 0;$

`while` $x < n$ `inv` $p = x * m$ ^ $x \leq n$ `do`

    $x := x + 1;$

    $p := p + m$

$\{p = n * m\}$

Example: VC Generation

Computing the guarded command

{ $n \geq 0$ }
assume $p_0 = p$; havoc p; assume $p = 0$;
assume $x_0 = x$; havoc x; assume $x = 0$;
assert p = x * m ∧ x ≤ n;
havoc x; havoc p; assume p = x * m ∧ x ≤ n;
((assume x < n;
assume x_1 = x; havoc x; assume x = x_1 + 1;
assume $p_1 = p$; havoc p; assume $p = p_1 + m$;
assert p = x * m ∧ x ≤ n; assume false)
|| assume x ≥ n)
{ $p = n * m$ }

Verification Condition Generation

Idea 1: Exhaustive symbolic execution of of GC program

- the program is correct if no assertion is ever falsified
- Verification Condition is constructed implicitly by symbolic exec
- Guided by pre-condition and program structure, but not guided by post-condition

Idea 2: propagate the post-condition backwards through the program:

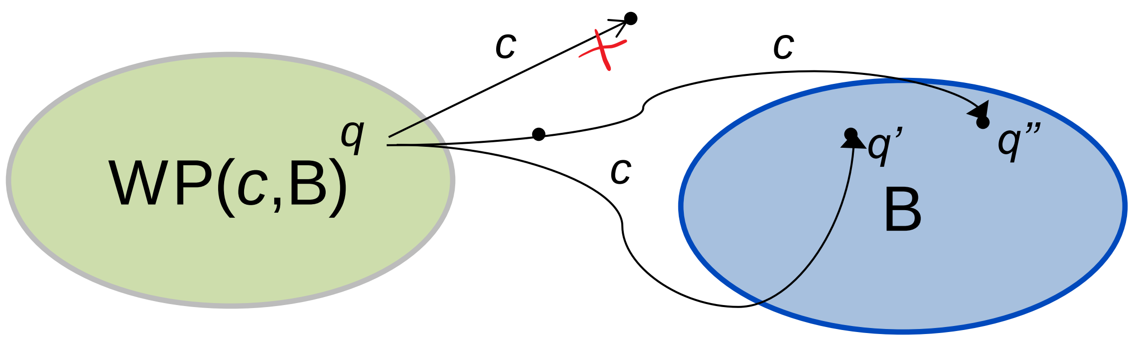
- From a Hoare triple {A} P {B}
- generate FOL formula $A \Rightarrow F(P, B)$
- Backwards propagation F(P, B) is formalized in terms of weakest preconditions.

UNIVERSITY OF WATERLOO

Weakest Preconditions

The weakest precondition WP(c,B) holds for any state q whose c-successor states all satisfy B:

$$q \vDash WP(c,B) \quad \text{iff} \quad \forall q' \in Q.\ q \xrightarrow{c} q' \Rightarrow q' \vDash B$$



Compute WP(P,B) recursively based on the structure of the program P.

Exercises: Weakest Pre-Condition

```
{                    } x := z + w { x >= y }
```

```
            {                         }
if y > 0 then x := x + 1 else x := y + 4
              {x >= 42}
```

```
            {                         }
if y > 0 then x := z else x := y
              {x >= y + z}
```

Exercises: Weakest Pre-Condition

```
{   z + w >= y   } x := z + w { x >= y }

            {                         }
if y > 0 then x := x + 1 else x := y + 4
                  {x >= 42}

            {                         }
if y > 0 then x := z else x := y
                {x >= y + z}
```

Exercises: Weakest Pre-Condition

```
{   z + w >= y   } x := z + w { x >= y }
```

```
{(y > 0 ∧ x >= 41) ∨ (y <= 0 ∧ y >= 38 )}
if y > 0 then x := x + 1 else x := y + 4
{x >= 42}
```

```
{          }
if y > 0 then x := z else x := y
{x >= y + z}
```

Exercises: Weakest Pre-Condition

```
{   z + w >= y   } x := z + w { x >= y }
```

```
{(y > 0 ∧ x >= 41) ∨ (y <= 0 ∧ y >= 38 )}
if y > 0 then x := x + 1 else x := y + 4
{x >= 42}
```

```
{(y > 0 ∧ z >= y + z) ∨ (y <= 0 ∧ y >= y + z)}
if y > 0 then x := z else x := y
{x >= y + z}
```

Exercises: Weakest Pre-Condition

```
{   z + w >= y   } x := z + w { x >= y }
```

```
{(y > 0 ∧ x >= 41) ∨ (y <= 0 ∧ y >= 38 )}
if y > 0 then x := x + 1 else x := y + 4
{x >= 42}
```

```
{(y > 0 ∧ 0 >= y) ∨ (y <= 0 ∧ 0 >= z)}
if y > 0 then x := z else x := y
{x >= y + z}
```

Computing Weakest Preconditions

WP(assume b, B) = b) B

WP(assert b, B) = b ^ B

WP(havoc x, B) = B[a/x] (*a* fresh in B) “replace x by *a*”

WP(c_1;c_2, B) = WP(c_1, WP(c_2, B))

WP(c_1 || c_2,B) = WP(c_1, B) ^ WP(c_2, B)

UNIVERSITY OF WATERLOO

Putting Everything Together

Given a Hoare triple H ´ {A} P {B}

Compute c_H = assume A; GC(P); assert B

Compute VC_H = WP(c_H, true)

Prove ` VC_H using a theorem prover.

UNIVERSITY OF WATERLOO

Example: VC Generation

Computing the weakest precondition

WP(assume $n \geq 0$;

assume $p_0 = p$; havoc p; assume $p = 0$;

assume $x_0 = x$; havoc x; assume $x = 0$;

assert $p = x * m \wedge x \leq n$;

havoc x; havoc p; assume $p = x * m \wedge x \leq n$;

((assume $x < n$;

assume $x_1 = x$; havoc x; assume $x = x_1 + 1$;

assume $p_1 = p$; havoc p; assume $p = p_1 + m$;

assert $p = x * m \wedge x \leq n$; assume false)

|| assume $x \geq n$) ;

assert $p = n * m$, true)

Example: VC Generation

Computing the weakest precondition

WP(assume $n \geq 0$;
assume $p_0 = p$; havoc p; assume $p = 0$;
assume $x_0 = x$; havoc x; assume $x = 0$;
assert $p = x * m \wedge x \leq n$;
havoc x; havoc p; assume $p = x * m \wedge x \leq n$;
(assume $x < n$;
assume $x_1 = x$; havoc x; assume $x = x_1 + 1$;
assume $p_1 = p$; havoc p; assume $p = p_1 + m$;
assert $p = x * m \wedge x \leq n$; assume false)
|| assume $x \geq n$, $p = n * m$)

UNIVERSITY OF WATERLOO

Example: VC Generation

Computing the weakest precondition

WP(assume $n \geq 0$;

assume $p_0 = p$; havoc p; assume $p = 0$;

assume $x_0 = x$; havoc x; assume $x = 0$;

assert $p = x * m \wedge x \leq n$;

havoc x; havoc p; assume $p = x * m \wedge x \leq n$,

WP((assume $x < n$;

assume $x_1 = x$; havoc x; assume $x = x_1 + 1$;

assume $p_1 = p$; havoc p; assume $p = p_1 + m$;

assert $p = x * m \wedge x \leq n$; assume false)

|| assume $x \geq n$, $p = n * m$))

Example: VC Generation

Computing the weakest precondition

$$
\begin{aligned}
\mathrm{WP}(\quad & \text{assume } n \geq 0; \\
& \text{assume } p_0 = p;\ \text{havoc } p;\ \text{assume } p = 0; \\
& \text{assume } x_0 = x;\ \text{havoc } x;\ \text{assume } x = 0; \\
& \text{assert } p = x * m \land x \leq n; \\
& \text{havoc } x;\ \text{havoc } p;\ \text{assume } p = x * m \land x \leq n, \\
& \ \mathrm{WP}(\text{assume } x < n; \\
& \quad \text{assume } x_1 = x;\ \text{havoc } x;\ \text{assume } x = x_1 + 1; \\
& \quad \text{assume } p_1 = p;\ \text{havoc } p;\ \text{assume } p = p_1 + m; \\
& \quad \text{assert } p = x * m \land x \leq n;\ \text{assume false},\ \ p = n * m) \\
& \land \mathrm{WP}(\text{assume } x \geq n,\ \ p = n * m))
\end{aligned}
$$

Example: VC Generation

Computing the weakest precondition

$$
\begin{aligned}
\mathrm{WP}(\;& \text{assume } n \geq 0;\\
& \text{assume } p_0 = p;\ \text{havoc } p;\ \text{assume } p = 0;\\
& \text{assume } x_0 = x;\ \text{havoc } x;\ \text{assume } x = 0;\\
& \text{assert } p = x * m \land x \leq n;\\
& \text{havoc } x;\ \text{havoc } p;\ \text{assume } p = x * m \land x \leq n,\\
& \mathrm{WP}(\text{assume } x < n;\\
& \quad \text{assume } x_1 = x;\ \text{havoc } x;\ \text{assume } x = x_1 + 1;\\
& \quad \text{assume } p_1 = p;\ \text{havoc } p;\ \text{assume } p = p_1 + m;\\
& \quad \text{assert } p = x * m \land x \leq n;\ \text{assume false},\ \ p = n * m)\\
& \land x \geq n \Rightarrow p = n * m)
\end{aligned}
$$

Example: VC Generation

Computing the weakest precondition

WP(assume $n \geq 0$;
assume $p_0 = p$; havoc p; assume $p = 0$;
assume $x_0 = x$; havoc x; assume $x = 0$;
assert $p = x * m \land x \leq n$;
havoc x; havoc p; assume $p = x * m \land x \leq n$,
WP (assume $x < n$;
assume $x_1 = x$; havoc x; assume $x = x_1 + 1$;
assume $p_1 = p$; havoc p; assume $p = p_1 + m$;
assert $p = x * m \land x \leq n$, WP(assume false, $p = n * m$)
$\land\, x \geq n \Rightarrow p = n * m$)

Example: VC Generation

Computing the weakest precondition

$$
\begin{array}{l}
\mathrm{WP}(\quad \text{assume } n \geq 0; \\
\quad \text{assume } p_0 = p;\ \text{havoc } p;\ \text{assume } p = 0; \\
\quad \text{assume } x_0 = x;\ \text{havoc } x;\ \text{assume } x = 0; \\
\quad \text{assert } p = x * m \land x \leq n; \\
\quad \text{havoc } x;\ \text{havoc } p;\ \text{assume } p = x * m \land x \leq n, \\
\quad \mathrm{WP}(\text{assume } x < n; \\
\quad\quad \text{assume } x_1 = x;\ \text{havoc } x;\ \text{assume } x = x_1 + 1; \\
\quad\quad \text{assume } p_1 = p;\ \text{havoc } p;\ \text{assume } p = p_1 + m; \\
\quad\quad \text{assert } p = x * m \land x \leq n,\ \text{false} \Rightarrow p = n * m) \\
\quad \land x \geq n \Rightarrow p = n * m)
\end{array}
$$

UNIVERSITY OF WATERLOO

Example: VC Generation

Computing the weakest precondition

$$
\begin{aligned}
WP(\quad & \text{assume } n \geq 0; \\
& \text{assume } p_0 = p;\ \text{havoc } p;\ \text{assume } p = 0; \\
& \text{assume } x_0 = x;\ \text{havoc } x;\ \text{assume } x = 0; \\
& \text{assert } p = x * m \land x \leq n; \\
& \text{havoc } x;\ \text{havoc } p;\ \text{assume } p = x * m \land x \leq n, \\
& \ WP(\text{assume } x < n; \\
& \quad \text{assume } x_1 = x;\ \text{havoc } x;\ \text{assume } x = x_1 + 1; \\
& \quad \text{assume } p_1 = p;\ \text{havoc } p;\ \text{assume } p = p_1 + m; \\
& \quad \text{assert } p = x * m \land x \leq n,\ \ \text{true}) \\
& \land x \geq n \Rightarrow p = n * m)
\end{aligned}
$$

Example: VC Generation

Computing the weakest precondition

WP(assume n ≥ 0;

assume $p_0 = p$; havoc p; assume $p = 0$;

assume $x_0 = x$; havoc x; assume $x = 0$;

assert $p = x * m \land x \leq n$;

havoc x; havoc p; assume $p = x * m \land x \leq n$,

WP (assume $x < n$;

assume $x_1 = x$; havoc x; assume $x = x_1 + 1$;

assume $p_1 = p$; havoc p; assume $p = p_1 + m$,

$p = x * m \land x \leq n$)

$\land\ x \geq n \Rightarrow p = n * m$)

UNIVERSITY OF WATERLOO

Example: VC Generation

Computing the weakest precondition

$$
\begin{aligned}
WP(\quad & \text{assume } n \geq 0; \\
& \text{assume } p_0 = p;\ \text{havoc } p;\ \text{assume } p = 0; \\
& \text{assume } x_0 = x;\ \text{havoc } x;\ \text{assume } x = 0; \\
& \text{assert } p = x * m \land x \leq n; \\
& \text{havoc } x;\ \text{havoc } p;\ \text{assume } p = x * m \land x \leq n, \\
& \ WP(\text{assume } x < n; \\
& \quad \text{assume } x_1 = x;\ \text{havoc } x;\ \text{assume } x = x_1 + 1; \\
& \quad \text{assume } p_1 = p;\ \text{havoc } p, \\
& \quad\ p = p_1 + m \Rightarrow p = x * m \land x \leq n) \\
& \land x \geq n \Rightarrow p = n * m)
\end{aligned}
$$

Example: VC Generation

Computing the weakest precondition

$WP($ $\text{assume } n \geq 0;$

$\text{assume } p_0 = p; \text{ havoc } p; \text{ assume } p = 0;$

$\text{assume } x_0 = x; \text{ havoc } x; \text{ assume } x = 0;$

$\text{assert } p = x * m \wedge x \leq n;$

$\text{havoc } x; \text{ havoc } p; \text{ assume } p = x * m \wedge x \leq n,$

$WP(\text{assume } x < n;$

$\text{assume } x_1 = x; \text{ havoc } x; \text{ assume } x = x_1 + 1,$

$p_1 = p \wedge pa_1 = p_1 + m \Rightarrow pa_1 = x * m \wedge x \leq n)$

$\wedge\, x \geq n \Rightarrow p = n * m)$

Example: VC Generation

Computing the weakest precondition

$$
\begin{aligned}
WP(\quad & \text{assume } n \geq 0; \\
& \text{assume } p_0 = p; \text{ havoc } p; \text{ assume } p = 0; \\
& \text{assume } x_0 = x; \text{ havoc } x; \text{ assume } x = 0; \\
& \text{assert } p = x * m \wedge x \leq n; \\
& \text{havoc } x; \text{ havoc } p; \text{ assume } p = x * m \wedge x \leq n, \\
& \; WP(\text{assume } x < n; \text{ assume } x_1 = x; \text{ havoc } x, \\
& \qquad x = x_1 + 1 \wedge p_1 = p \wedge pa_1 = p_1 + m \\
& \qquad\quad \Rightarrow pa_1 = x * m \wedge x \leq n) \\
& \wedge x \geq n \Rightarrow p = n * m)
\end{aligned}
$$

Example: VC Generation

Computing the weakest precondition

$$
\begin{aligned}
\mathrm{WP}(\quad & \text{assume } n \geq 0; \\
& \text{assume } p_0 = p;\ \text{havoc } p;\ \text{assume } p = 0; \\
& \text{assume } x_0 = x;\ \text{havoc } x;\ \text{assume } x = 0; \\
& \text{assert } p = x * m \land x \leq n; \\
& \text{havoc } x;\ \text{havoc } p;\ \text{assume } p = x * m \land x \leq n, \\
& \ \mathrm{WP}(\text{assume } x < n;\ \text{assume } x_1 = x, \\
& \quad xa_1 = x_1 + 1 \land\ p_1 = p \land\ pa_1 = p_1 + m \\
& \qquad \Rightarrow pa_1 = xa_1 * m \land xa_1 \leq n) \\
& \land x\ \geq n \Rightarrow p = n * m)
\end{aligned}
$$

Example: VC Generation

Computing the weakest precondition

$$
\begin{aligned}
WP(\quad & \text{assume } n \geq 0; \\
& \text{assume } p_0 = p;\ \text{havoc } p;\ \text{assume } p = 0; \\
& \text{assume } x_0 = x;\ \text{havoc } x;\ \text{assume } x = 0; \\
& \text{assert } p = x * m \land x \leq n; \\
& \text{havoc } x;\ \text{havoc } p;\ \text{assume } p = x * m \land x \leq n, \\
& WP(\text{assume } x < n, \\
& x_1 = x \land xa_1 = x_1 + 1 \land p_1 = p \land pa_1 = p_1 + m \\
& \qquad \Rightarrow pa_1 = xa_1 * m \land xa_1 \leq n) \\
& \land x \geq n \Rightarrow p = n * m)
\end{aligned}
$$

Example: VC Generation

Computing the weakest precondition

$$
\begin{aligned}
\mathrm{WP}(\quad & \text{assume } n \geq 0; \\
& \text{assume } p_0 = p;\ \text{havoc } p;\ \text{assume } p = 0; \\
& \text{assume } x_0 = x;\ \text{havoc } x;\ \text{assume } x = 0; \\
& \text{assert } p = x * m \land x \leq n; \\
& \text{havoc } x;\ \text{havoc } p;\ \text{assume } p = x * m \land x \leq n, \\
& \quad (x < n \land x_1 = x \land xa_1 = x_1 + 1 \land p_1 = p \land pa_1 = p_1 + m \\
& \qquad \Rightarrow pa_1 = xa_1 * m \land xa_1 \leq n) \\
& \land x \geq n \Rightarrow p = n * m)
\end{aligned}
$$

Example: VC Generation

Computing the weakest precondition

$n \geq 0 \land p_0 = p \land pa_3 = 0 \land x_0 = x \land xa_3 = 0 \Rightarrow pa_3 = xa_3 * m$
$\land\, xa_3 \leq n \,\land$

$(pa_2 = xa_2 * m \land xa_2 \leq n \Rightarrow$

$((xa_2 < n \land x_1 = xa_2 \land xa_1 = x_1 + 1 \land$
$p_1 = pa_2 \land pa_1 = p_1 + m) \Rightarrow pa_1 = xa_1 * m \land$
$xa_1 \leq n)$

$\land\, (xa_2 \geq n \Rightarrow pa_2 = n * m))$

Example: VC Generation

The resulting VC is equivalent to the conjunction of the following implications

$$n \geq 0 \land p_0 = p \land pa_3 = 0 \land x_0 = x \land xa_3 = 0 \Rightarrow$$
$$pa_3 = xa_3 * m \land xa_3 \leq n$$

$$n \geq 0 \land p_0 = p \land pa_3 = 0 \land x_0 = x \land xa_3 = 0 \land pa_2 = xa_2 * m \land$$
$$xa_2 \leq n \Rightarrow$$
$$xa_2 \geq n \Rightarrow pa_2 = n * m$$

$$n \geq 0 \land p_0 = p \land pa_3 = 0 \land x_0 = x \land xa_3 = 0 \land pa_2 = xa_2 * m \land xa_2 < n$$
$$\land\ x_1 = xa_2 \land xa_1 = x_1 + 1 \land p_1 = pa_2 \land pa_1 = p_1 + m \Rightarrow$$
$$pa_1 = xa_1 * m \land xa_1 \leq n$$

Example: VC Generation

simplifying the constraints yields

n ¸ 0) $0 = 0 * m$ ^ $0 \leq n$

$xa_2 \leq n$ ^ xa_2 ¸ n) $xa_2 * m = n * m$

$xa_2 <$ n) $xa_2 * m + m = (xa_2 + 1) * m$ ^ $xa_2 + 1 \leq n$

all of these implications are valid, which proves that the original Hoare triple was valid, too.

Software Verification