# Programming Paradigms

## 3rd Lecture

Prof. Janis Voigtländer

University of Duisburg-Essen

Summer Term 2017

# Important questions . . .

. . . which a syntax description (alone) cannot answer:

- What is the formal meaning of a (concrete) program?
- When are two programs equivalent?
- How can we prove what a program does?
- Can one determine the behavior of a program from that of its syntactic building blocks?
- Is it even possible to separately define the formal meaning of individual language constructs?
- When is a program correct?

# What does one actually mean by “correct”?

Partial correctness:

- Provision of input/output specifications:
    - Which inputs are needed?
    - What requirements do these inputs have to satisfy so that the program works?
    - Which outputs, with which properties, are produced?
- Every produced output satisfies the above output specification, if the inputs did satisfy the input specification.
- But it is not necessarily guaranteed that an output is always produced at all.

Total correctness: partial correctness + successful termination

# Hoare triples as verification formulas

We use "Hoare triples" as input/output specifications, notation:

$$\{\,P\,\}\ \mathsf{C}\ \{\,Q\,\} \qquad \text{or} \qquad \begin{array}{l}\{\,P\,\}\\ \mathsf{C}\\ \{\,Q\,\}\end{array}$$

where:

- C is a program piece (syntax), and
- $P$ and $Q$ are assertions about the state space of the (whole) program: logic expressions, containing program variables.

Beispiele: (not necessarily true propositions!)

- { x==0 } x=x+1; { x==1 }
- { (x==0) && (y==z) } x=1; x=2∗x; { (x>1) && ((y+z)!=3) }
- { x==0 } x=x+1; { x==0 }

# Hoare triples as verification formulas

Hoare triple $\{ P \}$ C $\{ Q \}$ to be read as:

- If in some program (memory) state the assertion $P$ holds,
- one then executes the program piece C and it terminates,
- then in the state afterwards the assertion $Q$ definitely holds.

(For concrete $P$, C, $Q$ what is said above can be true or not.)

Notes:

- This describes only partial correctness.
- The validity of $Q$ after execution of C is really relevant for every state in which $P$ holds[1].
- Hence: $P$ "alone" must be enough for the desired conclusion (so must be a strong enough assertion).
- But we are interested in the *weakest* assertion $P$ with that property.

[1] ... and in which C terminates ...

# Hoare triples as verification formulas

Some “special cases” and their meaning:

- $\{\, \mathit{true} \,\}\ \mathsf{C}\ \{\, Q \,\}$

  … means that whenever program piece C terminates, afterwards $Q$ holds (without any particular preconditions).

- $\{\, \mathit{false} \,\}\ \mathsf{C}\ \{\, Q \,\}$

  … means practically nothing. (Why?)

- $\{\, P \,\}\ \mathsf{C}\ \{\, \mathit{true} \,\}$

  … ditto.

- $\{\, P \,\}\ \mathsf{C}\ \{\, \mathit{false} \,\}$

  … means that starting from states in which $P$ holds, the execution of C never terminates.

# The Hoare calculus

. . . is meant to:

- make sure that assertions are used correctly,
- describe the transformation of assertions under execution of programs, and
- ultimately prove the truthfulness of verification formulas.

To this aim:

- formal capturing of the impact of basic language constructs on assertions
- rules to derive true verification formulas for larger programs from true verification formulas for program pieces

# Self-imposed restrictions

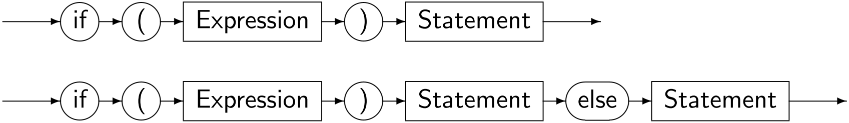
```
#include <stdio.h>

int main()
{ int n,s,i;
  scanf("%d",&n);
  s=0;
  i=1;
  while (i<=n)
    { s=s+i*i;
      i=i+1;
    }
  printf("%d",s);
  return 0;
}
```

We only consider such program pieces (particularly, no input/output, and only certain control structures)

# Rules for conditionals

Recall, from formal syntax description:



So when does $\{\,P\,\}$ **if** (T) C **else** D $\{\,Q\,\}$ hold?

- In every state in which $P$ holds, it could be the case that T holds, or not; depending on that then execution of C or D.
- In any case, afterwards $Q$ should hold.
- In the one case, we could express this requirement through the verification formula $\{\,P\,\&\&\,\mathrm{T}\,\}\ \mathrm{C}\ \{\,Q\,\}$, in the other through the verification formula $\{\,P\,\&\&\,!\mathrm{T}\,\}\ \mathrm{D}\ \{\,Q\,\}$.
- Since we need to be prepared for both cases, the following rule:

$$\frac{\{\,P\,\&\&\,\mathrm{T}\,\}\ \mathrm{C}\ \{\,Q\,\} \qquad \{\,P\,\&\&\,!\mathrm{T}\,\}\ \mathrm{D}\ \{\,Q\,\}}{\{\,P\,\}\ \textbf{if}\ (\mathrm{T})\ \mathrm{C}\ \textbf{else}\ \mathrm{D}\ \{\,Q\,\}}\ \mathrm{CR}$$

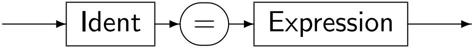
# Rules for conditionals

And what about $\{\,P\,\}$ **if** (T) C $\{\,Q\,\}$?

- Again, in every state in which $P$ holds, T either does or does not hold.
- In the one case, again reasonably require: $\{\,P\,\&\&\,\mathrm{T}\,\}$ C $\{\,Q\,\}$.
- But in the other case?
  - Since no execution of C (or of anything) in that case, simply require nothing additionally at all? Not a good idea!
  - Simply require $P \equiv Q$? Does not consider the "T-case"!
  - So require $(P\,\&\&\,!\mathrm{T}) \equiv Q$? Too strong!
  - Solution: require $(P\,\&\&\,!\mathrm{T}) \Rightarrow Q$!
    ("$\Rightarrow$" = logical implication, nothing to do with the Hoare calculus specifically)
- Hence, rule variant:

$$\frac{\{\,P\,\&\&\,\mathrm{T}\,\}\ \mathrm{C}\ \{\,Q\,\} \qquad (P\,\&\&\,!\mathrm{T}) \Rightarrow Q}{\{\,P\,\}\ \textbf{if}\ (\mathrm{T})\ \mathrm{C}\ \{\,Q\,\}}\ \mathrm{CR}$$

# Rules for assignment statements



In some sense the most simple kind of statement, but semantics surprisingly subtle.

First some examples of verification formulas that should be true:

1. { *true* } x=42; { x==42 }
2. { x==0 } x=x+1; { x==1 }
3. { y==0 } x=y+1; { x==1 }
4. { x==y } x=x+1; { x==y+1 }
5. { x!=y } z=x; { z!=y }
6. { x!=y } z=y; { x!=z }
7. { x!=y } z=y; { x!=y }

How could we capture all these cases in a uniform way, and do so by formulating a weakest precondition?

# Rules for assignment statements

A minimal (and actually sufficient) requirement to hold before an assignment x=e; so that afterwards $Q$ holds, is that (beforehand) the assertion $Q$ holds with all occurrences of x replaced by e.

Notation for the thus newly formed assertion: $Q^{\mathsf{x}}_{\mathsf{e}}$

Examples:

- $(\mathsf{x{=}{=}42})^{\mathsf{x}}_{\mathsf{42}} = (\mathsf{42{=}{=}42})$
- $(\mathsf{x{=}{=}1})^{\mathsf{x}}_{\mathsf{x+1}} = (\mathsf{x+1{=}{=}1})$
- $(\mathsf{x{=}{=}1})^{\mathsf{x}}_{\mathsf{y+1}} = (\mathsf{y+1{=}{=}1})$
- $(\mathsf{x!{=}z})^{\mathsf{z}}_{\mathsf{y}} = (\mathsf{x!{=}y})$
- $(\mathsf{x!{=}y})^{\mathsf{z}}_{\mathsf{y}} = (\mathsf{x!{=}y})$

# Rules for assignment statements

And indeed, it makes sense that:

1. { 42==42 } x=42; { x==42 }
2. { x+1==1 } x=x+1; { x==1 }
3. { y+1==1 } x=y+1; { x==1 }
4. . . .

Hence, reasonable rule (actually, an axiom):

$$\frac{}{\{\, Q^{\mathsf{x}}_{\mathsf{e}} \,\}\ \mathsf{x{=}e;}\ \{\, Q \,\}}\ \mathsf{AA}$$

However, we wanted to show 1. above under the precondition *true* (not under the precondition 42==42), as well as 2. under the precondition x==0 (not under the precondition x+1==1), etc.

Hence, rule variant:

$$\frac{P \Rightarrow Q^{\mathsf{x}}_{\mathsf{e}}}{\{\, P \,\}\ \mathsf{x{=}e;}\ \{\, Q \,\}}\ \mathsf{AA}$$

# Combination ⇝ Proof trees

Proof for “more complex” programs by plugging together individual rule applications:

$$\dfrac{\dfrac{\boxed{(\mathit{true}\,\&\&\,(\mathrm{x}<0)) \Rightarrow ((-\mathrm{x})>=0)}}{\begin{array}{l}\{\,\mathit{true}\,\&\&\,(\mathrm{x}<0)\,\}\\ \mathrm{x}=-\mathrm{x};\\ \{\,\mathrm{x}>=0\,\}\end{array}}\,\mathrm{AA} \qquad \boxed{(\mathit{true}\,\&\&\,!(\mathrm{x}<0)) \Rightarrow (\mathrm{x}>=0)}}{\{\,\mathit{true}\,\}\ \textbf{if}\ \ (\mathrm{x}<0)\ \mathrm{x}=-\mathrm{x};\ \{\,\mathrm{x}>=0\,\}}\,\mathrm{CR}$$

(AA = Assignment Axiom, CR = Conditional Rule)

Still open proof obligations (purely mathematical/logical expressions) are displayed in frames here, and from now on.

# Further useful rules

To "cut" larger program pieces (SR = Sequence Rule):

$$\frac{\{\,P\,\}\ \mathsf{C}\ \{\,R\,\} \qquad \{\,R\,\}\ \mathsf{D}\ \{\,Q\,\}}{\{\,P\,\}\ \mathsf{C\ D}\ \{\,Q\,\}}\ \mathsf{SR}$$

Potentially existing block markings are silently removed:

$$\frac{\{\,P\,\}\ \mathsf{C}\ \{\,Q\,\}}{\{\,P\,\}\ \{\mathsf{C}\}\ \{\,Q\,\}}\ \text{(often not even denoted in the tree)}$$

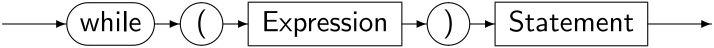
For "managing" pre- and postconditions
(SP = Stronger Precondition, WP = Weaker Postcondition):

$$\frac{P \Rightarrow R \qquad \{\,R\,\}\ \mathsf{C}\ \{\,Q\,\}}{\{\,P\,\}\ \mathsf{C}\ \{\,Q\,\}}\ \mathsf{SP}$$

$$\frac{\{\,P\,\}\ \mathsf{C}\ \{\,R\,\} \qquad R \Rightarrow Q}{\{\,P\,\}\ \mathsf{C}\ \{\,Q\,\}}\ \mathsf{WP}$$

# Key challenge: Dealing with loops

# For simplicity, only while-loops



while ( Expression ) Statement

When does $\{\, P \,\}$ **while** (T) C $\{\, Q \,\}$ hold?

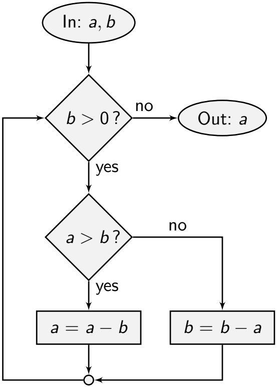
- As with **if**, we know that before (every) execution of program piece C here, the condition T holds.
- We also know that after finishing the loop (not just its body C), the condition T does not anymore hold.
- We know that during the first execution of the body C of the loop, beside T also $P$ holds.
- Unfortunately, we do not necessarily know that this is also the case during further executions of the body.
- If we allow ourselves the assumption, though, that C does not change the truth of $P$ (called loop invariant!), then:

here $P$ usually named as $Inv$

$$\frac{\{\, P \,\&\&\, \mathrm{T} \,\}\ \mathrm{C}\ \{\, P \,\}}{\{\, P \,\}\ \textbf{while}\ (\mathrm{T})\ \mathrm{C}\ \{\, P \,\&\&\, !\mathrm{T} \,\}}\ \text{IR (= Iteration Rule)}$$

# A concrete example

Let us consider:



respectively:

```
#include <stdio.h>

int main()
{ int a,b;
  scanf("%d",&a);
  scanf("%d",&b);
  while (b>0)
    { if (a>b) a=a-b;
      else b=b-a; }
  printf("%d",a);
  return 0; }
```

Verification goal:

{ (a==$A$) &&(b==$B$) &&(a>0) &&(b>=0) }
**while** (b>0) {**if** (a>b) a=a−b; **else** b=b−a;}
{ a==$gcd(A, B)$ }

# A concrete example

Verification goal:

$$\begin{array}{l}\{\,(\mathrm{a}{=}{=}A)\,\&\&(\mathrm{b}{=}{=}B)\,\&\&(\mathrm{a}>0)\,\&\&(\mathrm{b}>{=}0)\,\}\\ \textbf{while}\ (\mathrm{b}>0)\ \{\textbf{if}\ (\mathrm{a}>\mathrm{b})\ \mathrm{a}{=}\mathrm{a}-\mathrm{b};\ \textbf{else}\ \mathrm{b}{=}\mathrm{b}-\mathrm{a};\}\\ \{\,\mathrm{a}{=}{=}gcd(A,B)\,\}\end{array}$$

Obviously, we will need to apply the iteration rule:

$$\frac{\{\,Inv\ \&\&\ \mathrm{T}\,\}\ \mathrm{C}\ \{\,Inv\,\}}{\{\,Inv\,\}\ \textbf{while}\ (\mathrm{T})\ \mathrm{C}\ \{\,Inv\ \&\&\ !\mathrm{T}\,\}}\ \mathrm{IR}$$

Since $\mathrm{a}{=}{=}gcd(A,B)$ does not cover $!(\mathrm{b}>0)$, we need to add (at least) that, via the rule for weaker postcondition:

$$\dfrac{\dfrac{\{\,Inv\,\&\&(\mathrm{b}>0)\,\}\ \ \ldots\ \ \{\,Inv\,\}}{\begin{array}{l}\{\,Inv\,\}\\ \textbf{while}\ (\mathrm{b}>0)\ \{\ldots\}\\ \{\,Inv\ \&\&\ !(\mathrm{b}>0)\,\}\end{array}}\ \mathrm{IR} \qquad \boxed{(Inv\ \&\&\ !(\mathrm{b}>0)) \Rightarrow (\mathrm{a}{=}{=}gcd(A,B))}}{\{\,\ldots\,\}\ \textbf{while}\ (\mathrm{b}>0)\ \{\ldots\}\ \{\,\mathrm{a}{=}{=}gcd(A,B)\,\}}\ \mathrm{WP}$$

# A concrete example

But the loop invariant cannot simply be the originally given $P$, which was: $(\mathrm{a}{=}{=}A)\,\&\&(\mathrm{b}{=}{=}B)\,\&\&(\mathrm{a}>0)\,\&\&(\mathrm{b}>{=}0)$. (Why?)

Hence, also application of the rule for stronger precondition:

$$\dfrac{\boxed{P \Rightarrow \mathit{Inv}} \qquad \dfrac{\{\, \mathit{Inv}\,\&\&(\mathrm{b}>0)\,\}\ \ \ldots\ \ \{\, \mathit{Inv}\,\}}{\begin{array}{l}\{\, \mathit{Inv}\,\}\\ \textbf{while}\ (\mathrm{b}>0)\ \{\ldots\}\\ \{\, \mathit{Inv}\,\&\&\ !(\mathrm{b}>0)\,\}\end{array}}\,\mathrm{IR}}{\{\, P\,\}\ \textbf{while}\ (\mathrm{b}>0)\ \{\ldots\}\ \{\, \mathit{Inv}\,\&\&\ !(\mathrm{b}>0)\,\}}\,\mathrm{SP}$$

So the “only” remaining problem now is to find *Inv* such that:

1. $$\dfrac{\vdots}{\{\, \mathit{Inv}\,\&\&(\mathrm{b}>0)\,\}\ \textbf{if}\ (\mathrm{a}>\mathrm{b})\ \mathrm{a}{=}\mathrm{a}-\mathrm{b};\ \textbf{else}\ \mathrm{b}{=}\mathrm{b}-\mathrm{a};\ \{\, \mathit{Inv}\,\}}$$
2. $\boxed{P \Rightarrow \mathit{Inv}}$
3. $\boxed{(\mathit{Inv}\,\&\&\ !(\mathrm{b}>0)) \Rightarrow (\mathrm{a}{=}{=}\mathit{gcd}(A,B))}$

# A concrete example

Idea: Exploit that the *gcd* of a and b does not change when one subtracts one from the other.

So, *Inv* could be: $(\mathit{gcd}(\mathrm{a},\mathrm{b}){=}{=}\mathit{gcd}(A,B))\,\&\&(\mathrm{a}>0)\,\&\&(\mathrm{b}>{=}0)$

check that 2. and 3. hold!

To then establish the required

$$\frac{\vdots}{\{\,\mathit{Inv}\,\&\&(\mathrm{b}>0)\,\}\ \textbf{if}\ \ (\mathrm{a}>\mathrm{b})\ \mathrm{a}{=}\mathrm{a}-\mathrm{b};\ \textbf{else}\ \mathrm{b}{=}\mathrm{b}-\mathrm{a};\ \{\,\mathit{Inv}\,\}}$$

which is still open, first an application of the conditional rule:

$$\frac{\begin{array}{l}\{\,\mathit{Inv}\,\&\&(\mathrm{b}>0)\,\&\&(\mathrm{a}>\mathrm{b})\,\}\\ \mathrm{a}{=}\mathrm{a}-\mathrm{b};\\ \{\,\mathit{Inv}\,\}\end{array}\qquad\begin{array}{l}\{\,\mathit{Inv}\,\&\&(\mathrm{b}>0)\,\&\&\ !(\mathrm{a}>\mathrm{b})\,\}\\ \mathrm{b}{=}\mathrm{b}-\mathrm{a};\\ \{\,\mathit{Inv}\,\}\end{array}}{\{\,\mathit{Inv}\,\&\&(\mathrm{b}>0)\,\}\ \textbf{if}\ \ (\mathrm{a}>\mathrm{b})\ \mathrm{a}{=}\mathrm{a}-\mathrm{b};\ \textbf{else}\ \mathrm{b}{=}\mathrm{b}-\mathrm{a};\ \{\,\mathit{Inv}\,\}}\ \mathrm{CR}$$

# A concrete example

... and then in both branches an assignment axiom on top:

$$\frac{\boxed{(\mathit{Inv}\,\&\&(\mathrm{b}>0)\,\&\&(\mathrm{a}>\mathrm{b})) \Rightarrow \mathit{Inv}^{\mathrm{a}}_{\mathrm{a}-\mathrm{b}}}}{\{\,\mathit{Inv}\,\&\&(\mathrm{b}>0)\,\&\&(\mathrm{a}>\mathrm{b})\,\}\ \mathrm{a}{=}\mathrm{a}-\mathrm{b};\ \{\,\mathit{Inv}\,\}}\ \mathrm{AA}$$

and

$$\frac{\boxed{(\mathit{Inv}\,\&\&(\mathrm{b}>0)\,\&\&\ !(\mathrm{a}>\mathrm{b})) \Rightarrow \mathit{Inv}^{\mathrm{b}}_{\mathrm{b}-\mathrm{a}}}}{\{\,\mathit{Inv}\,\&\&(\mathrm{b}>0)\,\&\&\ !(\mathrm{a}>\mathrm{b})\,\}\ \mathrm{b}{=}\mathrm{b}-\mathrm{a};\ \{\,\mathit{Inv}\,\}}\ \mathrm{AA}$$

Due to *Inv* being $(\mathit{gcd}(\mathrm{a},\mathrm{b}){=}{=}\mathit{gcd}(A,B))\,\&\&(\mathrm{a}>0)\,\&\&(\mathrm{b}>=0)$,

- $\mathit{Inv}^{\mathrm{a}}_{\mathrm{a}-\mathrm{b}}$ is:
  $(\mathit{gcd}(\mathrm{a}-\mathrm{b},\mathrm{b}){=}{=}\mathit{gcd}(A,B))\,\&\&(\mathrm{a}-\mathrm{b}>0)\,\&\&(\mathrm{b}>=0)$
- $\mathit{Inv}^{\mathrm{b}}_{\mathrm{b}-\mathrm{a}}$ is:
  $(\mathit{gcd}(\mathrm{a},\mathrm{b}-\mathrm{a}){=}{=}\mathit{gcd}(A,B))\,\&\&(\mathrm{a}>0)\,\&\&(\mathrm{b}-\mathrm{a}>=0)$

The proof obligations still to prove (see above) do indeed hold!

# A concrete example: Complete proof tree

$$
\dfrac{
  \boxed{P \Rightarrow \mathit{Inv}}
  \quad
  \dfrac{
    \dfrac{
      \dfrac{\boxed{(\mathit{Inv}\ \&\&(b>0)\ \&\&(a>b)) \Rightarrow \mathit{Inv}^{a}_{a-b}}}{\begin{array}{l}\{\ \mathit{Inv}\ \&\&(b>0)\ \&\&(a>b)\ \}\\ a=a-b;\\ \{\ \mathit{Inv}\ \}\end{array}}\ \text{AA}
      \quad
      \dfrac{\boxed{(\mathit{Inv}\ \&\&(b>0)\ \&\&\ !(a>b)) \Rightarrow \mathit{Inv}^{b}_{b-a}}}{\begin{array}{l}\{\ \mathit{Inv}\ \&\&(b>0)\ \&\&\ !(a>b)\ \}\\ b=b-a;\\ \{\ \mathit{Inv}\ \}\end{array}}\ \text{AA}
    }{\begin{array}{l}\{\ \mathit{Inv}\ \&\&(b>0)\ \}\\ \textbf{if}\ \ (a>b)\ a=a-b;\ \textbf{else}\ b=b-a;\\ \{\ \mathit{Inv}\ \}\end{array}}\ \text{CR}
  }{\begin{array}{l}\{\ \mathit{Inv}\ \}\\ \textbf{while}\ (b>0)\ \{\textbf{if}\ \ (a>b)\ a=a-b;\ \textbf{else}\ b=b-a;\}\\ \{\ \mathit{Inv}\ \&\&\ !(b>0)\ \}\end{array}}\ \text{IR}
}{\begin{array}{l}\{\ P\ \}\\ \textbf{while}\ (b>0)\ \{\textbf{if}\ \ (a>b)\ a=a-b;\ \textbf{else}\ b=b-a;\}\\ \{\ \mathit{Inv}\ \&\&\ !(b>0)\ \}\end{array}}\ \text{SP}
$$

$$
\dfrac{
  \begin{array}{l}\{\ P\ \}\\ \textbf{while}\ (b>0)\ \{\textbf{if}\ \ (a>b)\ a=a-b;\ \textbf{else}\ b=b-a;\}\\ \{\ \mathit{Inv}\ \&\&\ !(b>0)\ \}\end{array}
  \quad
  \boxed{(\mathit{Inv}\ \&\&\ !(b>0)) \Rightarrow (a==\mathit{gcd}(A,B))}
}{\begin{array}{l}\{\ P\ \}\\ \textbf{while}\ (b>0)\ \{\textbf{if}\ \ (a>b)\ a=a-b;\ \textbf{else}\ b=b-a;\}\\ \{\ a==\mathit{gcd}(A,B)\ \}\end{array}}\ \text{WP}
$$

where

$P$ is: $(a==A)\ \&\&(b==B)\ \&\&(a>0)\ \&\&(b>=0)$

$\mathit{Inv}$ is: $(\mathit{gcd}(a,b)==\mathit{gcd}(A,B))\ \&\&(a>0)\ \&\&(b>=0)$

# Summary of the Hoare calculus rules

$$\frac{\{\,P\;\&\&\;\mathrm{T}\,\}\;\mathrm{C}\;\{\,Q\,\}\qquad\{\,P\;\&\&\;!\mathrm{T}\,\}\;\mathrm{D}\;\{\,Q\,\}}{\{\,P\,\}\;\textbf{if}\;\;(\mathrm{T})\;\mathrm{C}\;\textbf{else}\;\mathrm{D}\;\{\,Q\,\}}\;\mathrm{CR}$$

$$\frac{\{\,P\;\&\&\;\mathrm{T}\,\}\;\mathrm{C}\;\{\,Q\,\}\qquad(P\;\&\&\;!\mathrm{T})\Rightarrow Q}{\{\,P\,\}\;\textbf{if}\;\;(\mathrm{T})\;\mathrm{C}\;\{\,Q\,\}}\;\mathrm{CR}$$

$$\frac{}{\{\,Q^{\mathrm{x}}_{\mathrm{e}}\,\}\;\mathrm{x{=}e;}\;\{\,Q\,\}}\;\mathrm{AA}\qquad\qquad\frac{P\Rightarrow Q^{\mathrm{x}}_{\mathrm{e}}}{\{\,P\,\}\;\mathrm{x{=}e;}\;\{\,Q\,\}}\;\mathrm{AA}$$

$$\frac{\{\,P\,\}\;\mathrm{C}\;\{\,R\,\}\qquad\{\,R\,\}\;\mathrm{D}\;\{\,Q\,\}}{\{\,P\,\}\;\mathrm{C}\;\mathrm{D}\;\{\,Q\,\}}\;\mathrm{SR}$$

$$\frac{P\Rightarrow R\qquad\{\,R\,\}\;\mathrm{C}\;\{\,Q\,\}}{\{\,P\,\}\;\mathrm{C}\;\{\,Q\,\}}\;\mathrm{SP}\qquad\frac{\{\,P\,\}\;\mathrm{C}\;\{\,R\,\}\qquad R\Rightarrow Q}{\{\,P\,\}\;\mathrm{C}\;\{\,Q\,\}}\;\mathrm{WP}$$

$$\frac{\{\,Inv\;\&\&\;\mathrm{T}\,\}\;\mathrm{C}\;\{\,Inv\,\}}{\{\,Inv\,\}\;\textbf{while}\;\;(\mathrm{T})\;\mathrm{C}\;\{\,Inv\;\&\&\;!\mathrm{T}\,\}}\;\mathrm{IR}$$

# Application to another example

```
#include <stdio.h>

 int main()
 { int n,s,i;
   scanf("%d",&n);
   s=0;
   i=1;
   while (i<=n)
     { s=s+i*i;
       i=i+1;
     }
   printf("%d",s);
   return 0;
 }
```

Example run:

- n==3, s==0, i==1
- n==3, s==1, i==1
- n==3, s==1, i==2
- n==3, s==5, i==2
- n==3, s==5, i==3
- n==3, s==14, i==3
- n==3, s==14, i==4

Verification goal:

{ (n>=0) && (s==0) && (i==1) }
**while** (i<=n) {s=s+i∗i; i=i+1;}
{ s== $\sum_{j=1}^{n} j^2$ }

# Application to another example

Verification goal:

$$\begin{array}{l} \{\,(\mathsf{n}>=0)\,\&\&(\mathsf{s}==0)\,\&\&(\mathsf{i}==1)\,\} \\ \textbf{while}\ (\,\mathsf{i}<=\mathsf{n})\ \{\mathsf{s}=\mathsf{s}+\mathsf{i}*\mathsf{i};\ \mathsf{i}=\mathsf{i}+1;\} \\ \{\,\mathsf{s}==\sum_{j=1}^{\mathsf{n}} j^2\,\} \end{array}$$

Again, as in previous example, use of SP and WP rules, towards:

$$\dfrac{\dfrac{\vdots}{\{\,\mathit{Inv}\,\&\&\,(\,\mathsf{i}<=\mathsf{n})\,\}\ \mathsf{s}=\mathsf{s}+\mathsf{i}*\mathsf{i};\ \ \mathsf{i}=\mathsf{i}+1;\ \{\,\mathit{Inv}\,\}}}{\{\,\mathit{Inv}\,\}\ \textbf{while}\ (\,\mathsf{i}<=\mathsf{n})\ \{\mathsf{s}=\mathsf{s}+\mathsf{i}*\mathsf{i};\ \mathsf{i}=\mathsf{i}+1;\}\ \{\,\mathit{Inv}\,\&\&\,!(\,\mathsf{i}<=\mathsf{n})\,\}}\ \mathsf{IR}$$

Where for the still to determine *Inv* it should hold that:

1. $\boxed{((\mathsf{n}>=0)\,\&\&(\mathsf{s}==0)\,\&\&(\mathsf{i}==1)) \Rightarrow \mathit{Inv}}$
2. $\boxed{(\mathit{Inv}\,\&\&\,!(\,\mathsf{i}<=\mathsf{n})) \Rightarrow (\mathsf{s}==\sum_{j=1}^{\mathsf{n}} j^2)}$

# Application to another example

Where for the still to determine *Inv* it should hold that:

1. $\boxed{((\mathrm{n}{>}{=}0)\,\&\&(\mathrm{s}{=}{=}0)\,\&\&(\mathrm{i}{=}{=}1)) \Rightarrow \mathit{Inv}}$

2. $\boxed{(\mathit{Inv}\;\&\&\;!(\mathrm{i}{<}{=}\mathrm{n})) \Rightarrow (\mathrm{s}{=}{=}\sum_{j=1}^{\mathrm{n}} j^2)}$

To determine the loop invariant, recall:

- $\underline{\mathrm{n}{=}{=}3,\ \mathrm{s}{=}{=}0,\ \mathrm{i}{=}{=}1}$
- n==3, s==1, i==1
- $\underline{\mathrm{n}{=}{=}3,\ \mathrm{s}{=}{=}1,\ \mathrm{i}{=}{=}2}$
- n==3, s==5, i==2
- $\underline{\mathrm{n}{=}{=}3,\ \mathrm{s}{=}{=}5,\ \mathrm{i}{=}{=}3}$
- n==3, s==14, i==3
- $\underline{\mathrm{n}{=}{=}3,\ \mathrm{s}{=}{=}14,\ \mathrm{i}{=}{=}4}$

Aha!
*Inv* is: $(0{<}\mathrm{i}{<}{=}\mathrm{n}{+}1)\,\&\&(\mathrm{s}{=}{=}\sum_{j=1}^{\mathrm{i}-1} j^2)$

(and that even satisfies 1. and 2.)

# Application to another example

So what remains to establish:

$$\frac{\vdots}{\{\, \mathit{Inv} \,\&\&\, (\mathsf{i}<=\mathsf{n}) \,\}\ \mathsf{s=s+i*i;\ \ i=i+1;}\ \{\, \mathit{Inv} \,\}}$$

with *Inv* being $(0<\mathsf{i}<=\mathsf{n}+1)\,\&\&(\mathsf{s}==\sum_{j=1}^{\mathsf{i}-1} j^2)$

Twice assignment axiom (before that, sequence rule):

$$\dfrac{\dfrac{\boxed{(\mathit{Inv} \,\&\&\, (\mathsf{i}<=\mathsf{n})) \Rightarrow (\mathit{Inv}\,\big|^{\mathsf{i}}_{\mathsf{i}+1})^{\mathsf{s}}_{\mathsf{s+i*i}}}}{\{\, \mathit{Inv} \,\&\&\, (\mathsf{i}<=\mathsf{n}) \,\}\ \mathsf{s=s+i*i;}\ \{\, \mathit{Inv}\,\big|^{\mathsf{i}}_{\mathsf{i}+1} \,\}}\,\mathsf{AA} \qquad \dfrac{}{\{\, \mathit{Inv}\,\big|^{\mathsf{i}}_{\mathsf{i}+1} \,\}\ \mathsf{i=i+1;}\ \{\, \mathit{Inv} \,\}}\,\mathsf{AA}}{\{\, \mathit{Inv} \,\&\&\, (\mathsf{i}<=\mathsf{n}) \,\}\ \mathsf{s=s+i*i;\ \ i=i+1;}\ \{\, \mathit{Inv} \,\}}\,\mathsf{SR}$$

Remains to check: $\boxed{\begin{array}{l} ((0<\mathsf{i}<=\mathsf{n})\,\&\&(\mathsf{s}==\sum_{j=1}^{\mathsf{i}-1} j^2)) \\ \Rightarrow ((0<\mathsf{i}+1<=\mathsf{n}+1)\,\&\&(\mathsf{s+i*i}==\sum_{j=1}^{\mathsf{i}} j^2)) \end{array}}$ ✓

# Application to another example: Complete proof tree

$$
\dfrac{
  \boxed{((n>=0)\,\&\&(s==0)\,\&\&(i==1)) \Rightarrow \mathit{Inv}}
  \qquad
  \dfrac{
    \dfrac{
      \dfrac{
        \boxed{(\mathit{Inv}\ \&\&\ (i<=n)) \Rightarrow (\mathit{Inv}\big|^{i}_{i+1})^{s}_{s+i*i}}
      }{
        \begin{array}{l}
        \{\ \mathit{Inv}\ \&\&\ (i<=n)\ \}\\
        s=s+i*i;\\
        \{\ \mathit{Inv}\big|^{i}_{i+1}\ \}
        \end{array}
      }\,\text{AA}
      \qquad
      \dfrac{}{
        \begin{array}{l}
        \{\ \mathit{Inv}\big|^{i}_{i+1}\ \}\\
        i=i+1;\\
        \{\ \mathit{Inv}\ \}
        \end{array}
      }\,\text{AA}
    }{
      \begin{array}{l}
      \{\ \mathit{Inv}\ \&\&\ (i<=n)\ \}\\
      s=s+i*i;\ \ i=i+1;\\
      \{\ \mathit{Inv}\ \}
      \end{array}
    }\,\text{SR}
  }{
    \begin{array}{l}
    \{\ \mathit{Inv}\ \}\\
    \textbf{while}\ (i<=n)\ \{s=s+i*i;\ i=i+1;\}\\
    \{\ \mathit{Inv}\ \&\&\ !(\ i<=n)\ \}
    \end{array}
  }\,\text{IR}
}{
  \begin{array}{l}
  \{\ (n>=0)\,\&\&(s==0)\,\&\&(i==1)\ \}\\
  \textbf{while}\ (i<=n)\ \{s=s+i*i;\ i=i+1;\}\\
  \{\ \mathit{Inv}\ \&\&\ !(\ i<=n)\ \}
  \end{array}
}\,\text{SP}
$$

$$
\dfrac{
  \begin{array}{l}
  \{\ (n>=0)\,\&\&(s==0)\,\&\&(i==1)\ \}\\
  \textbf{while}\ (i<=n)\ \{s=s+i*i;\ i=i+1;\}\\
  \{\ \mathit{Inv}\ \&\&\ !(\ i<=n)\ \}
  \end{array}
  \qquad
  \boxed{(\mathit{Inv}\ \&\&\ !(\ i<=n)) \Rightarrow (s==\sum_{j=1}^{n} j^2)}
}{
  \begin{array}{l}
  \{\ (n>=0)\,\&\&(s==0)\,\&\&(i==1)\ \}\\
  \textbf{while}\ (i<=n)\ \{s=s+i*i;\ i=i+1;\}\\
  \{\ s==\sum_{j=1}^{n} j^2\ \}
  \end{array}
}\,\text{WP}
$$

where

$\mathit{Inv}$ is: $(0<i<=n+1)\,\&\&(s==\sum_{j=1}^{i-1} j^2)$