

1 PROGRAM SEMANTICS

The semantics of a programming language, P, assigns meaning to Programs written in that language. But what is the meaning of such a valid string? The meaning of "meaning" is a question long considered by philosophers. One appealing view (Nozick The Examined Life) considers the meaning of X to be its **connection to something else** which is of value, where possession of integrated structure gives value. That is close to the view we take of the meaning of programming languages.

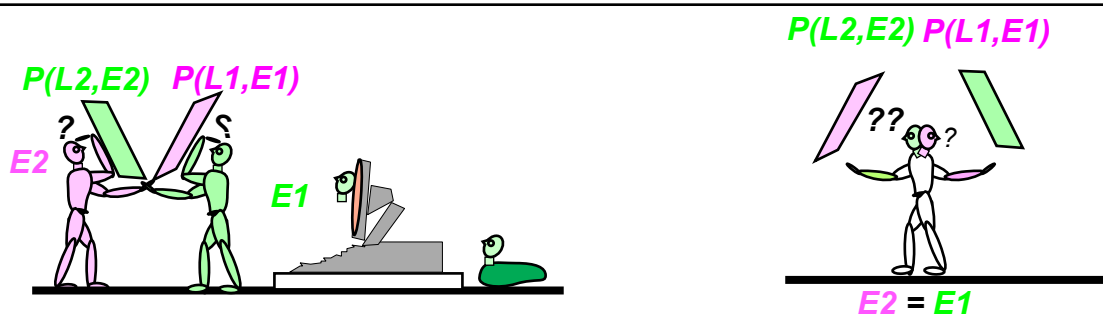
We view a Program's semantics as the translation of P1 into another language to give a program or an assertion or set of assertions, P2 which is better understood than P1, by some target entity. There are well established ways of doing this for some cases-Specifically:

1. Basic elements of P1 can be declared equivalent to basic elements of P2, and then the Composition of these basic elements of P1 can be associated with the corresponding composition basic elements of P2. Then P1 can be analysed into its basic components and the translation used to produce the composition of the corresponding basic components of P2. This might be called **Translational Semantics (no Proofs)**.

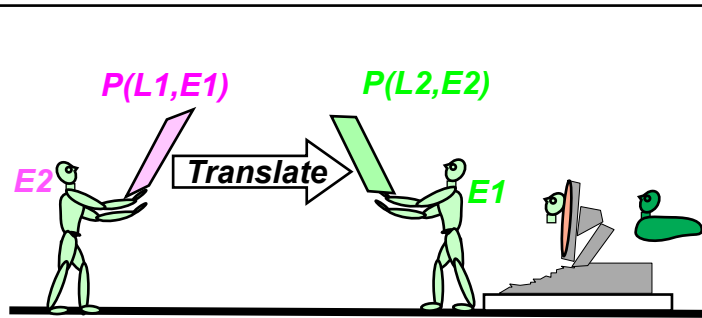
This is the technique used in **Attribute Grammar Translations**. This is generally applicable to translation from a higher level language to a lower level language and is a technique for synthesizing that L2 from L1. L1 being perhaps better understood by the programmer and L2 being better understood by a machine. The attribute grammar is not easily inverted, so the inverse, lower to higher translation is not easily obtained.

2. Given a program in P1 written in L1 one can assert that at completion it will have some effect expressed as P2 in a language L2. In this case one cannot synthesise P2 from P1 because P2 is a higher level than P1. (that is the corresponding basic structures of the language cannot be easily composed in similar ways).

In this case we need to assert in L2 what we expect to be the case at strategic parts of L1 and then an analysis starting with the P2 expectation corresponding to the effect at the end of P1-and determine what must be true in L2 terms at the beginning of P1. If nothing need be true then the expectation that P2 is true at the beginning is justified this verification approach is called **Axiomatic Semantics.(Proofs)**



Consider $P(L1,E1)$ a Program written in language, L1 whose meaning is clear to a person, animal or machine (an entity), $E1$; but less clear to an entity $E2$. $E2$ would understand the same program better if written in language L2, namely $P(L2,E2)$. When $E1$ and $E2$ are different entities the situation is represented in figure (a). But it may be that $E1 = E2$ (the same entity) and $P(L1,E1)$ and $P(L2,E2)$ are different programs of different clearness to $E1=E2$ as illustrated in figure (b). So $P(E1,L1) = \text{meaning} = P(E2,L2)$, but the understanding to different entities of each differ



For the case that $P(L1,E1)$ is the program produced in a higher level language (JAVA, C++) and $P(L2,E2)$ is in lower level language (ex. assembly or machine language) there is available the Attribute Grammar which allows the compact, precise description of the necessary translation, as well as the implementation of that translation. If however $P(L1,E1)$ is a lower language level and $P(L2,E2)$ is the higher level language then one needs to invert an Attribute Grammar of $P(L2,E2)$ to $P(L1,E1)$ a translation generally difficult to implement.

1.1 Program Verification

General form: $\{P_0\} \langle \text{statement} \rangle \{Q_1\}$, P_0 and Q_1 are each a set of assertions

P_0 's assertions are true at time 0,

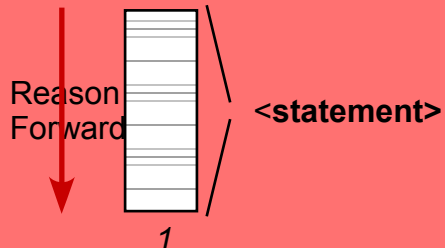
References to a variable X , is to X 's value at time 0 just before $\langle \text{statement} \rangle$, designated X_0 .

Q_1 's assertions are true immediately after $\langle \text{statement} \rangle$ as a result of its execution and P_0 .

Reference to variable X in Q_1 , is to its *new* value at time 1, just before $\langle \text{statement} \rangle$, designated X_1 .

There two questions that can be approached

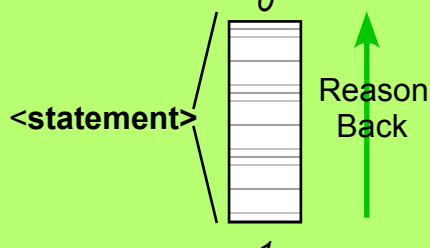
Given P_0 is known to be true before Program



This Program Results in Q_1 being true

1. **Forward:** If P_0 is true at time 0 and $\langle \text{statement} \rangle$ is then executed and finishes at time 1, what consequently is true at time 1, i.e., what is Q_1 ?

The Expectation: is fulfilled provided P_0 is true



Expectation: This Program Results in Q_1 being true

Back: If Q_1 is true at time 1 as a result of $\langle \text{statement} \rangle$, whose execution start and finished at time 1, what must have been true at time 0, what was P_0 ?

$P(L1,E1)$

$X \leftarrow -5$
 $Y \leftarrow -2^X$

P is $\{2^5 == 32\}$

$X \leftarrow -5$

$\{2^X == 32\}$

$Y \leftarrow -2^X$

Q is $\{Y == 32\}$

$P(L2,E2)$

Q is $\{Y == 32\}$

So running $P(L1,E1)$ we assert results in Q is $\{Y == 32\}$ P is $\{2^5 == 32\}$ is true before $P(L1,E1)$

$P(L1,E1)$

$X \leftarrow -a;$
 $Y \leftarrow -b$
 $T \leftarrow -X$
 $X \leftarrow -Y$
 $Y \leftarrow -T$

P is $b == b \& a == a$

$X \leftarrow -a;$

$b == b \& X == a$

$Y \leftarrow -b$

$Y == b \& X == a$

$T \leftarrow -X$

$Y == b \& T == a$

$X \leftarrow -Y$

$X == b \& T == a$

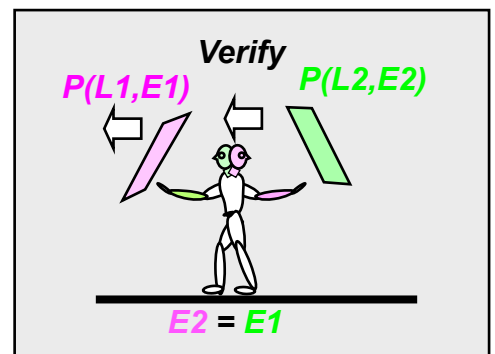
$Y \leftarrow -T$

Q is $X == b \& Y == a$

$P(L2,E2)$

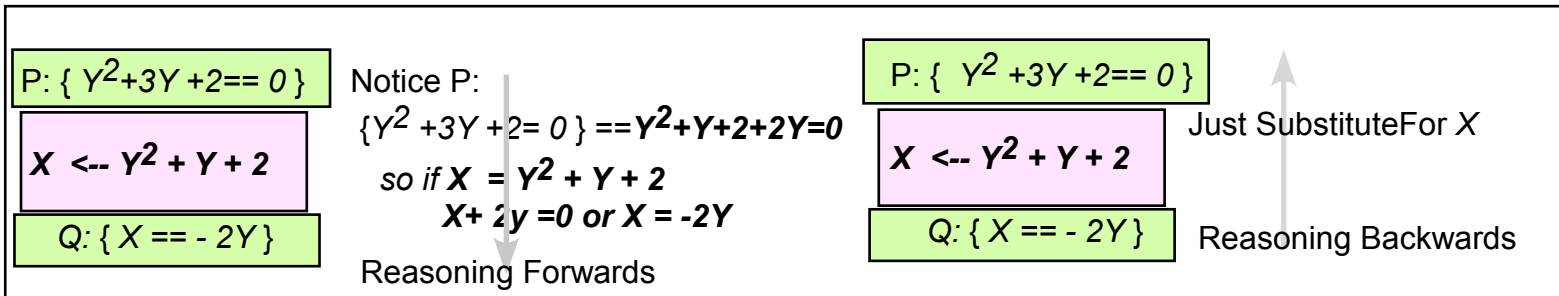
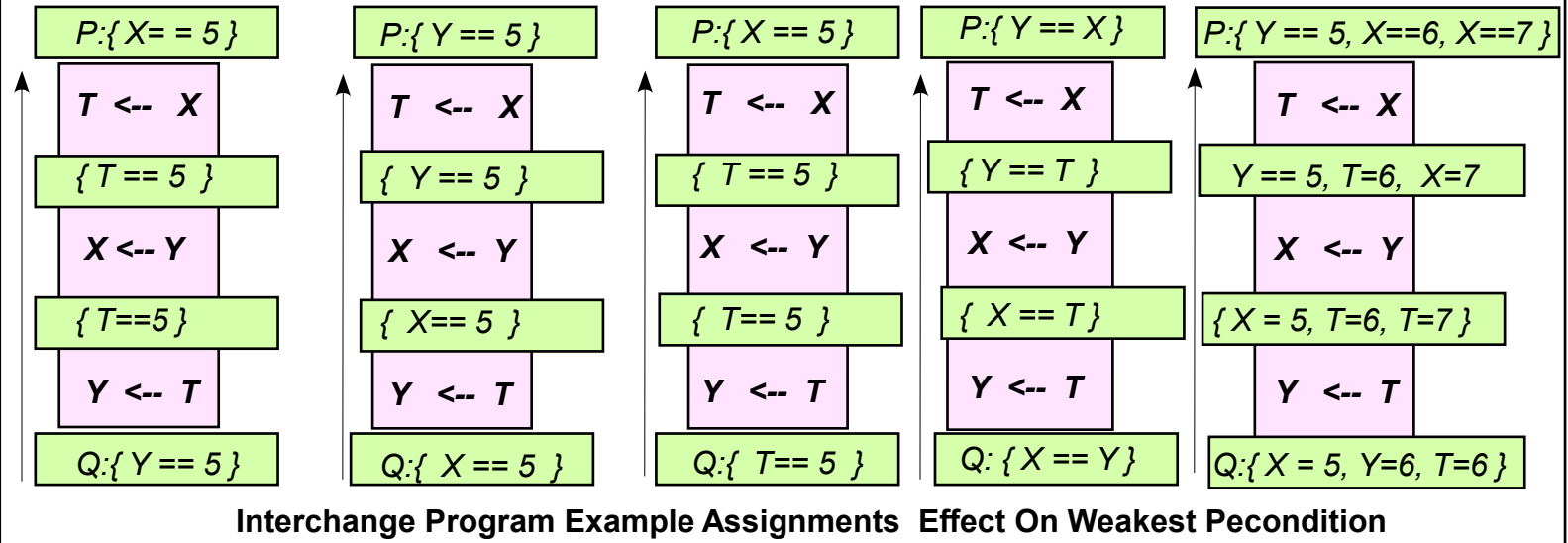
Q is $X == b \& Y == a$

So running $P(L1,E1)$ we assert results in Q is $X == b \& Y == a$ and the precondition for this to be true is P is $b == b \& a == a$ which is clearly true before $P(L1,E1)$



ASSIGNMENT STATEMENTS EXAMPLES

Q IS TRUE AFTER PROGRAM IFF P IS TRUE BEFORE PROGRAM.
 IF P IS UNIVERSALLY TRUE/FALSE THEN IF THE PROGRAM IS RUN Q WILL BE TRUE/FALSE
 INDEPENDENT OF VARIABLE VALUES BEFORE PROGRAM.

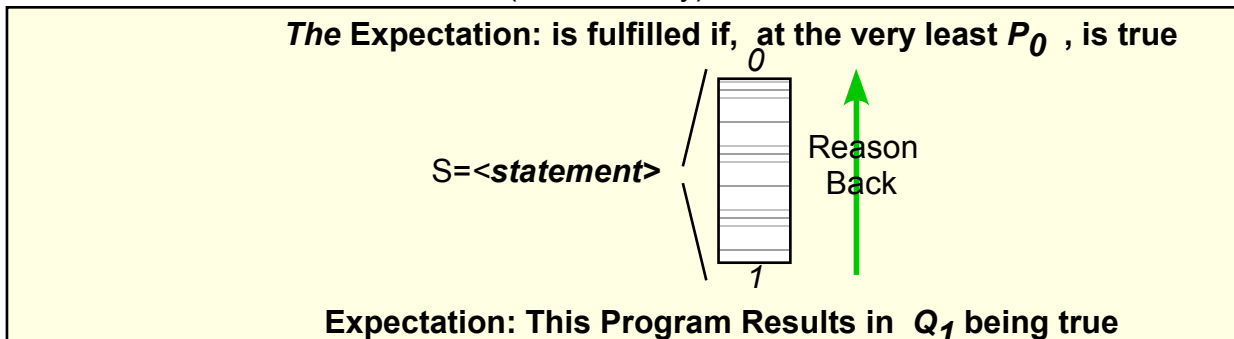


1.2 The Weakest Precondition

The definition in 1.1 does not sufficiently restrict the meaning of P and Q . It allows either to contain any number of truths which are completely independent of the statement execution. It is the **minimal** set of assertions, P whose truth before **<statement>** is guaranteed to result in the set of truths, Q , after the statement, P should be a weak, in fact the weakest, precondition for Q . This is expressed as

$$P = wp(\langle \text{statement} \rangle, Q).$$

Consider two assertions A and B where $A \Rightarrow B$ (implies) B . Then we say B is as weak or weaker than A . meaning B is true whenever A is true, though the converse is not assured. If C is a precondition, and for all D such that $C \Rightarrow D$, D is not a precondition, then C is the weakest precondition. The weakest precondition P **must** be satisfied before **<statement>** (is necessary) for Q to be true after **<statement>**



In general we would like to be able to ascertain the weakest precondition, P , given any Q , and any **<statement>** where **<statement>** is anything from a program fragment to an entire program. In the case of an entire program we presumably know the assertions we want to be true at the end namely-“what the program is to accomplish”. Our objective is to find under what initial condition these will be true: none should be necessary. Beyond verification, determination of the weakest precondition is sometimes useful in the design of a program. If the desired result, say Q , of a part, say S , of a program, P , is known then the precondition shows what has to be made true before S in P if Q is to be true after S .

2. STATEMENT TYPES AND VERIFICATION

For each of the basic statement types the relation between the pre- and post-conditions are given. For a program using these statements the relation of its pre and post condition is obtained by applying the relation frompre to post condition for each statement one after the other.

2.1 Assignment Statements

Next we give the general approach for getting assertions after and before assignments. If the **< statement >** is an assignment:

$$\{ P(X_0, Y_0, Z_0) \} X_1 \leftarrow f(X_0, Y_0, Z_0) \{ Q(X_1, Y_1, Z_1) \}$$

(The subscripts are not part of the name of the variables. They identify the relative time at which reference is made to their value.)

Then the only variable that could have changed value between times 0 and 1 is X. For any other variable, W; $W_1 = W_0$. So at time 1 the following is true:

1. $Y_1 = Y_0$
2. $Z_1 = Z_0$
3. $X_1 = f(X_0, Y_0, Z_0)$

Therefore From:
 $Q(X_1, Y_1, Z_1)$

We get

$$Q(f(X_0, Y_0, Z_0), Y_0, Z_0) = P(X_0, Y_0, Z_0)$$

So we can represent the effect of an assignment as follows:

$$\{ P(X_0, Y_0, Z_0) \} = \{ Q(f(X_0, Y_0, Z_0), Y_0, Z_0) \} X_1 \leftarrow f(X_0; Y_0; Z_0) \{ Q(X_1, Y_1, Z_1) \}$$

This can be generalized to more than one simultaneously applied variable assignments

$$\{ P(X_0, Y_0, Z_0) \} = \{ Q(f(X_0, Y_0; Z_0), g(X_0, Y_0, Z_0), Z_0) \} \left| \begin{array}{l} X_1 \leftarrow f(X_0, Y_0, Z_0); \\ Y_1 \leftarrow g(X_0, Y_0, Z_0); \end{array} \right| \{ Q(X_1, Y_1, Z_1) \}$$

or dropping the subscripts for the single assignment:

$$\{ P(X, Y, Z) \} = \{ Q(f(X, Y, Z), Y, Z) \} X \leftarrow f(X, Y, Z) \{ Q(X, Y, Z) \}$$

and for the double assignment:

$$\{ P(X_0, Y_0, Z_0) \} = \{ Q(f(X, Y, Z), g(X, Y, Z), Z) \} \left| \begin{array}{l} X \leftarrow f(X, Y, Z); \\ Y \leftarrow g(X, Y, Z); \end{array} \right| \{ Q(X, Y, Z) \}$$

$$\text{wp} \left(\begin{array}{l} X_1 \leftarrow f(X_0, Y_0, Z_0) \\ Y_1 \leftarrow g(X_0, Y_0, Z_0) \end{array} \middle| Q(X_1, Y_1, Z_1) \right) = Q(f(X_0, Y_0, Z_0), g(X_0, Y_0, Z_0), Z_0)$$

$$\left| \begin{array}{l} X_1 \leftarrow f(X_0, Y_0, Z_0) \\ Y_1 \leftarrow g(X_0, Y_0, Z_0) \end{array} \right| \{ Q(X_1, Y_1, Z_1) \}$$

These results can be used to reason from Q to P (backward) by substituting $f(X; Y; Z)$ for X throughout Q.

To reason from P to Q (forward) substituting X for $f(X; Y; Z)$ throughout P is adequate in simple cases.

we have seen thus far. More generally: to move forward from P through an assignment $X \leftarrow f(X; Y; Z)$;

1. solve the equation $X = f(X; Y; Z)$ for X and substitute the result for X throughout P. (This can be difficult or impossible and may result in a number of equally valid solutions.

2. In addition all relations not involving X should also be placed in Q since they were not changed by the assignment.

The forward reasoning is difficult (as illustrated on the next page) and fortunately not required.

2.2 The If Statement

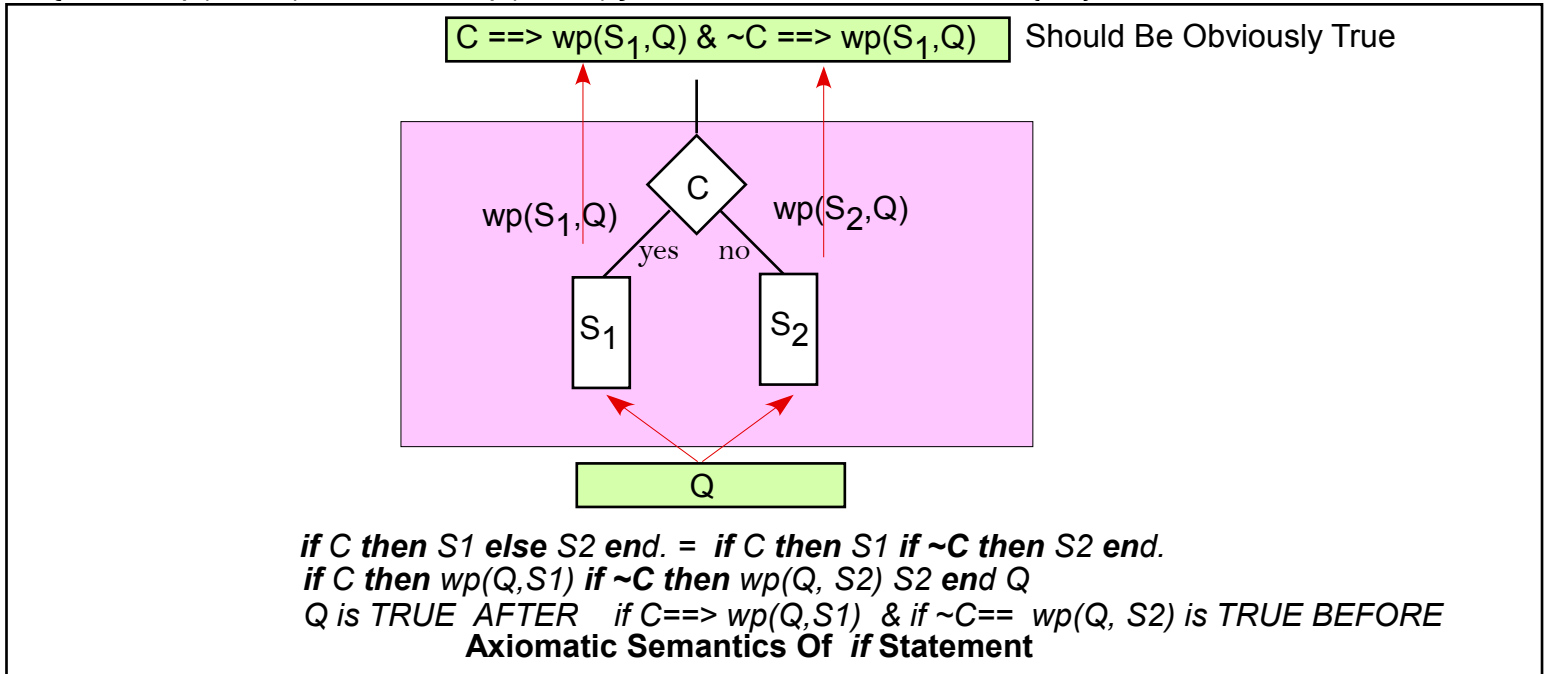
To handle this statement forward reasoning is combined with the weakest precondition backward reasoning to show the weakest precondition, P , before the if execution which will insure the expected results for each branch of the if

Consider an if function of the form:

if C then S1 else S2 end.

Suppose that Q is true after, and P is true before the if operation: Then, if C is true $S1$ will be executed so the weakest precondition $wp(S1, Q)$ must be true before the if in order that Q would be true after. On the otherhand for similar reason if $\sim C$ holds $wp(S2, Q)$ must hold before the if..

$\{ C \implies wp(S1; Q) \ \& \ \sim C \implies wp(S2; Q) \}$ **if C then S1 else S2 end** $\{ Q \}$

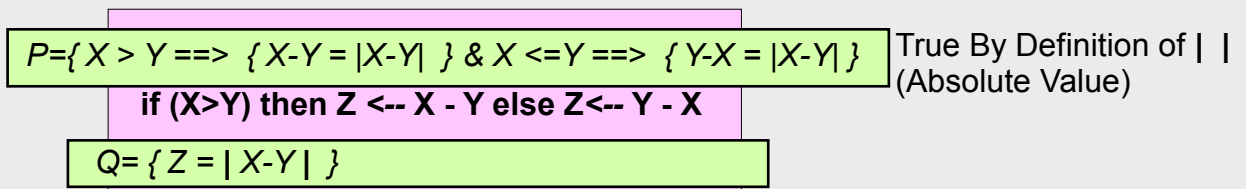


$P(L1, E1)$

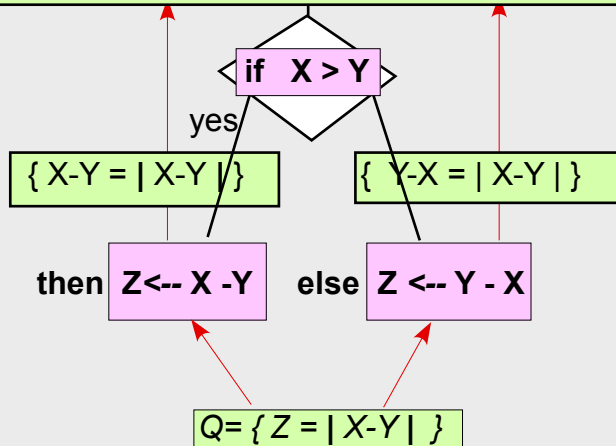
if (X>Y) then Z \leftarrow X - Y else Z \leftarrow Y - X

$P(L2, E2)$

$Q = \{ Z = |X - Y| \}$



$\{ X > Y \implies \{ X - Y = |X - Y| \} \ \& \ X \leq Y \implies \{ Y - X = |X - Y| \} \}$



Computing Absolute Value of Difference Of X and Y

$P(L1,E1)$

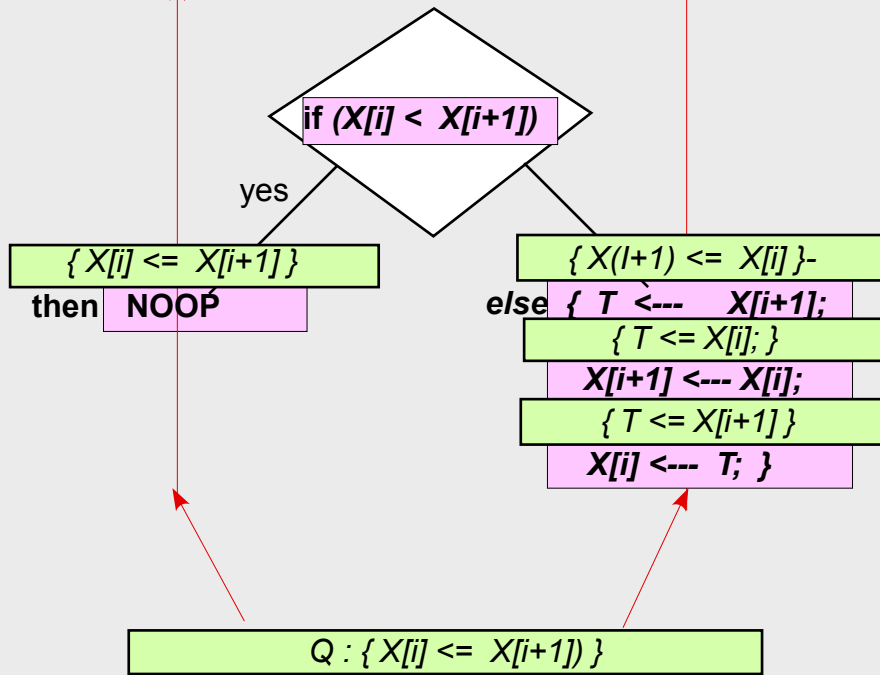
$P(L2,E2)$

if $(X[i] < X[i+1])$ then NOOP else { $T \leftarrow X[i+1]; X[i+1] \leftarrow X[i]; X[i] \leftarrow T;$ }

$Q : \{ X[i] \leq X[i+1] \}$

$\{ X[i] < X[i+1] \} \implies X[i] \leq X[i+1] \ \& \ [X[i] \geq X[i+1] \implies X[i+1] \leq X[i] \}$
 if $(X[i] < X[i+1])$ then NOOP else { $T \leftarrow X[i+1]; X[i+1] \leftarrow X[i]; X[i] \leftarrow T;$ }
 $Q : \{ X[i] \leq X[i+1] \}$

$\{ X[i] < X[i+1] \} \implies X[i] \leq X[i+1] \ \& \ [X[i] \geq X[i+1] \implies X[i+1] \leq X[i] \}$ Obviously True



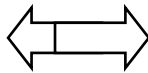
Basic Step In Interchange Sort-Proof

To show how an assertion about the function performed by a while can be proved. This requires formulating that assertion in a special way called an invariant. This is an assertion A, (an Invariant) held to be true after the body of the while whose weakest precondition at the start of the body of the while together with the condition for remaining in the while is also A. This formulation can be tricky. Instead we will find a recursive definition of a function whose call has the same result as the while. This can allow the replacement of the while with an conditional assignment statement. Then we can prove the expected outcome or property of the outcome in which we are interested is correct. (Later we also show some examples of the use of Invariants)

In our first example

RECURSIVE FUNCTION DEFINITION

- 1) $f(R,D) = 1 + f(R-D,D)$ if $(R \geq D)$
- 2) $f(R,D) = 0$ if $(R < D)$



WHILE FUNCTION

```
F=0;
while(R>=D)
{ R =R-D;
  F=F+1; }
```

if $(R \geq D)$ $F = \lfloor R/D \rfloor$ otherwise $F=0$

Proposed Solution??:

$f(R,D) = \lfloor R/D \rfloor$, if $R \geq D$ otherwise $=0$
 $\lfloor R/D \rfloor$ signifies integer division of R by D
 {that is for this f:
 $f(P_1,P_2) = \lfloor P_1/P_2 \rfloor$,

PROOF THAT SOLUTION IS VALID

- 1) $\lfloor R/D \rfloor = 1 + \lfloor (R-D)/D \rfloor$ if $(R \geq D)$
 - 1) $\lfloor R/D \rfloor = 1 + \lfloor (R-D-D)/D \rfloor$ if $(R \geq D)$
 - 1) $\lfloor R/D \rfloor = 1 + \lfloor (R-D-1) \rfloor$ if $(R \geq D)$
 - 1) $\lfloor R/D \rfloor = 1 + \lfloor (R/D) - 1 \rfloor$ if $(R \geq D)$
 - 1) $\lfloor R/D \rfloor = \lfloor (R/D) \rfloor$ if $(R \geq D)$ OK!
- 2) $\lfloor R/D \rfloor = 0$ if $(R < D)$ OK!

So, With $f(R,D)$ defined as above the assignment $F = f(A,B)$ is equivalent to the closed form assignment $F = \lfloor R/D \rfloor$

SEMANTICS OF RECURSIVE FUNCTION

A STRAIGHT FORWARD EVALUATION OF RECURSIVE DEFINITION:

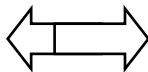
$f(R,D) = 1 + f(R-D,D)$ if $(R-D \geq D)$

$f(R,D) = 1 + 1 + f(R-2D,D)$ if $(R-2D \geq D)$

...

$f(R,D) = 1 + 1 + \dots + 1 + f(R-nD)$ if $(R-nD \geq D)$

$f(R,D) = \underbrace{1 + 1 + \dots + 1}_n + 0 = n$ if $(R-nD < D)$



A STRAIGHT FORWARD EVALUATION OF WHILE DEFINITION:

if $(R \geq D)$ $R = R-D$
 $F = 0 + 1 = 1$

if $(R \geq 2D)$ $R = R-D-D = R-2D$
 $F = 1 + 1 = 2$

...

if $(R \geq D)$ $R = R-D-D- \dots -D = R-nD$
 $F = \underbrace{1 + 1 + \dots + 1}_n = n$

if $(R < D)$ $F = \underbrace{1 + 1 + \dots + 1}_n = n$

PROVING CLOSED FORM SATISFIES RECURSIVE FUNCTION DEFINITION

Therefore the WHILE FUNCTION can be replaced by the assignment statement $F = \lfloor R/D \rfloor$,

THE WHILE STATEMENT EQUIVALENT TAIL RECURSION

Converted To An Assignment Statement
 $F =$ The Number Of times D can be subtracted from R and $R-FD \geq D$, i.e., $F = \lfloor R/D \rfloor$

RELATION OF RECURSIVE DEFINITION AND WHILE STATEMENT

1] $\text{rev}(L) = []$ if $L = []$
 2] $\text{rev}(L) = \text{append}(\text{rev}(\text{cdr}(L)), \text{list}(\text{car}(L)))$ if $L \neq []$

Another way to write this is

DEF:

1] $\text{rev}(L) = []$ if $L = []$
 2] $\text{rev}(L=[x_1, x_2, \dots, x_n]) = \text{append}(\text{rev}(\text{cdr}(L=[x_1, x_2, \dots, x_n])), \text{list}(\text{car}(L=[x_1, x_2, \dots, x_n])))$ if $L=[x_1, x_2, \dots, x_n]$ if $L \neq []$

We claim that: $\text{rev}(L=[x_1, x_2, \dots, x_n]) = [x_n, \dots, x_2, x_1]$.

Assuming we know what append, car, and list do, the proof is obtained by substituting for $\text{rev}([x_1, x_2, \dots, x_n])$ in DEF and verifying that the equalities hold.

PROOF:

1] $[] = ? []$ yes

2] $[x_n, \dots, x_2, x_1] = ? \text{append}(\text{rev}(\text{cdr}(L) = [x_n, \dots, x_2]), \text{list}(x_1)) = [x_1]$ yes
 $[x_n, \dots, x_2, x_1] = ! [x_n, \dots, x_2, x_1]$ yes

DEF:

1] $f(x) = f(x-1) + 2x$ if $x > 1$
 2] $f(1) = 2$

We claim $f(x) = x^2 + x$

PROOF:

1] $x^2 + x = ? (x-1)^2 + (x-1) + 2x = (x^2 - 2x + 1 + (x-1) + 2x) = x^2 + x!$

2] $1^2 + 1 = ? 2!$

SEMANTICS OF RECURSIVE FUNCTION DEFINITIONS

1] $f(n) = n + f(n-1)$ if $n > 1$
 2] $f(1) = 1$ if $n=1$

Proposed Solution=Fixed Point $f(n) = n(n+1)/2$

1] $n(n+1)/2 =? n + (n)(n-1)/2$ if $n > 0$
 $=? [2n + n^2 - n]/2$ if $n > 0$
 $=! n(n+1)/2$ if $n > 0$

2] $1(2)/2 =? 1$ if $n=1$
 $1 =! 1$ if $n=1$

$\{1/.5 < .5\}$

Z=.5

$\{1/(Z-1) < Z\}$
 $\{Z / Z(Z-1) < Z\}$

X= (Z(Z-1)) originally $X= 2f(Z)$

$\{Z / X < Z\}$

Y= Z/X

$\{Y < Z\}$

1] $f(n) = n + f(n-1)$ if $n > 0$
 2] $f(1) = 1$

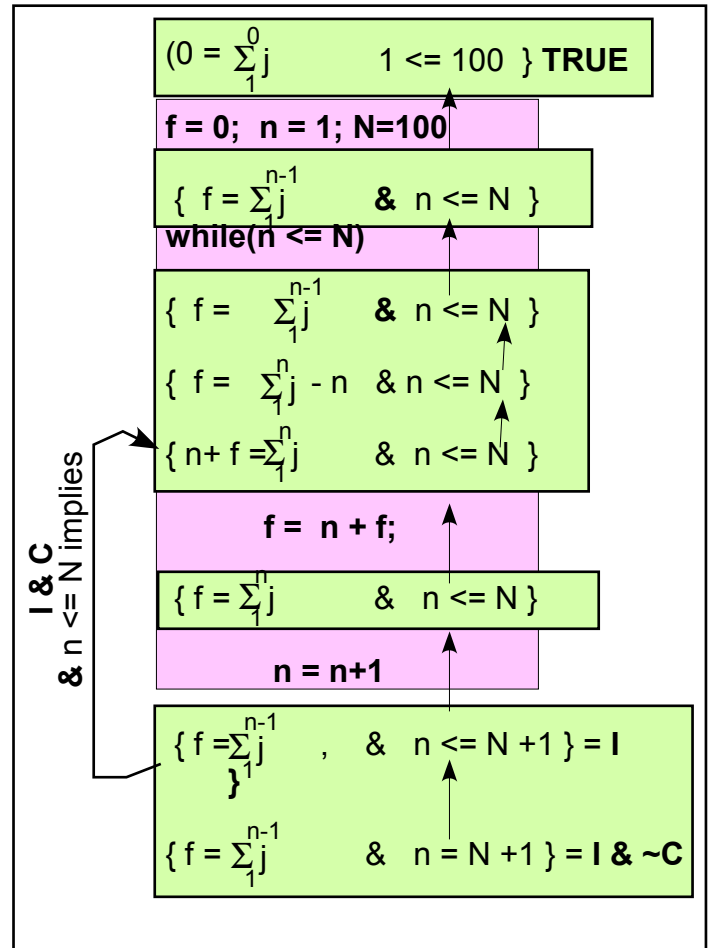
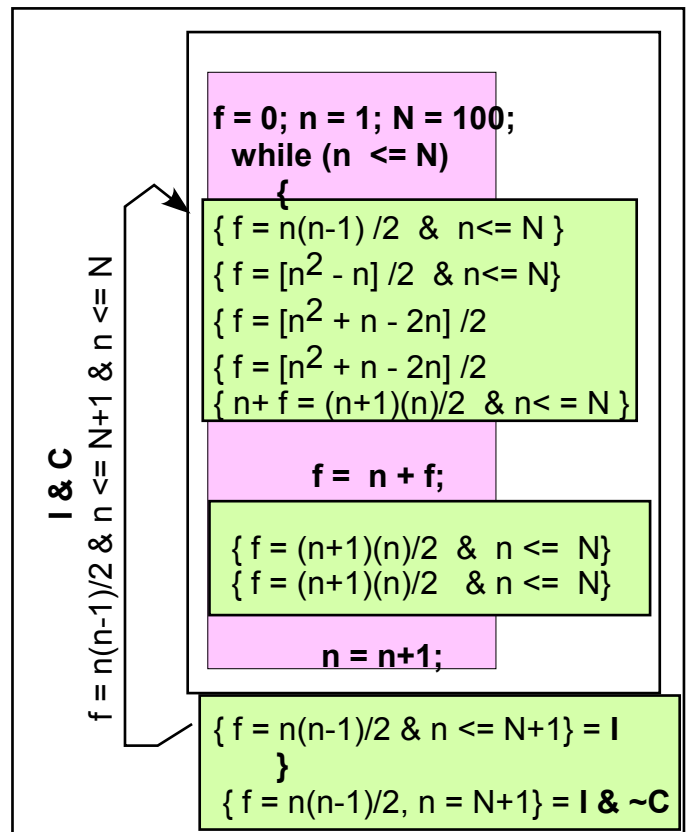
Proposed Solution=Fixed Point $f(n) = \sum j$

1] $\sum_1^n j =? n + \sum_1^{n-1} j$ if $n > 0$

$\sum_1^n j =! \sum_1^n j$ if $n > 0$

$\sum_1^1 j = 1$ if $n = 0$

2] $1 =! 1$ if $n=0$



SEMANTICS OF RECURSIVE FUNCTION DEFINITIONS

AN CAUTIONARY EXAMPLE

$\{(\text{intX} + \text{intW} + \text{intY} + \text{intZ}) / 4 = \text{average}(\text{intX}, \text{intY}, \text{intZ}, \text{intW}) : \text{TRUE BY DEFINITION-Quite}$

X = int X; Y = intY; W = int W

$\{(X + W + Y + Z) / 4 = \text{average}(X+W - W, Y, Z, W) = \text{average}(X, Y, Z, W) \text{ TRUE BY DEFINITION}$

X = X + W

$\{(X + Y + Z) / 4 = \text{average}(X-W, Y, Z, W) \}$

A = X + Y

$\{(A + Z) / 4 = \text{average}(X-W, Y, Z, W) \}$

A = A + Z

$\{ A / 4 = \text{average}(X-W, Y, Z, W) \}$

A = A/4

$\{ A = \text{average}(X - W, Y, Z, W) \}$ this statement is asserted to be true of the variable values after the 4 line program above is run.

ONLY IF $\{(X + W + Y + Z) / 4 = \text{average}(X, Y, Z, W) \}$ IS TRUE BEFORE THE PROGRAM RUNS
IS $\{ A / 4 = \text{average}(X-W, Y, Z, W) \}$ TRUE AFTER IT RUNS

$\{(\text{intX} + \text{intW} + \text{intY} + \text{intZ}) / 4 = \text{average}(\text{intX}, \text{intY}, \text{intZ}, \text{intW}) : \text{TRUE BY DEFINITION-Quite}$

X = int X; Y = intY W = int W

$\{(X + W + Y + Z) / 4 = \text{average}(X, Y, Z, W) \text{ TRUE BY DEFINITION?}$

A = X + W

$\{(A + Y + Z) / 4 = \text{average}(X, Y, Z, W) \}$

A = A + Y

$\{(A + Z) / 4 = \text{average}(X-W, Y, Z, W) \}$

A = A + Z

$\{ A / 4 = \text{average}(X, Y, Z, W) \}$

A = A/4

$\{ A = \text{average}(X, Y, Z, W) \}$ this statement is asserted to be true of the variable values after the 4 line program above is run.

ONLY IF $\{(X + W + Y + Z) / 4 = \text{average}(X, Y, Z, W) \}$ IS TRUE BEFORE THE PROGRAM RUNS
IS $\{ A / 4 = \text{average}(X, Y, Z, W) \}$ TRUE AFTER IT RUNS

A/4 has the same value at the end both examples.

Yet in (a) A= the average of X, Y, Z and W and the other A is the average of (X-W), Y, Z, W

$$g(F,R,D) = g(F+1,D-D) \quad R \geq D \quad \text{if } R \geq D > 0$$

$$g(F,R,D) = F \quad R < D \quad \text{if } D > R > 0$$

$$g(F,R,D) == F + \lfloor \lfloor R/D \rfloor \rfloor$$

$$K == F_0 + \lfloor R_0/D_0 \rfloor \quad R_0 \geq 0, D_0 > 0$$

$$R=R_0, D=D_0, F=F_0;$$

while($R \geq D$)

$$K == F+1 + \lfloor (R-D)/D \rfloor, \quad R-D \geq 0, D > 0 \rightarrow K = F + R/D \quad R \geq 0, D > 0$$

{ $R = R-D$;

$$K = F+1 + \lfloor R/D \rfloor, \quad R \geq 0, D > 0$$

$F=F+1$;

$$K == F + \lfloor R/D \rfloor, \quad R \geq 0, D > 0$$

}

$$K == F + \lfloor R/D \rfloor, \quad D > R > 0 \quad D > 0$$

$$K == F$$

$$F_0 + \lfloor R_0/D_0 \rfloor == F + \lfloor R/D \rfloor \quad D > R > 0 \quad D > 0$$