**Tutorial Nov 23**

* ex:

proc main() {

assert(X + 1 = Y + 2)

X := X + 1;

assert(X = Y + 2) % asn

Y := Y + 2;

assert(X = Y) % asn

}

* ex:

proc main() {

assert(true)

if (C > B) then {

assert(C > B) % if-then-else

assert(C ≥ B ∧ C ≥ C) % implied (VC 1)

M := C;

assert(M ≥ B ∧ M ≥ C) % asn

} else {

assert(C ≤ B) % if-then-else

assert(B ≥ B ∧ B ≥ C) % implied (VC 2)

M := B;

assert(M ≥ B ∧ M ≥ C) % asn

};

assert(M ≥ B ∧ M ≥ C) % if-then-else

}

VC 1: C > B |− C ≥ B ∧ C ≥ C

1) C > B premise

2) C ≥ B by arith % on 1

3) C ≥ C by arith

4) C ≥ B ∧ C ≥ C by and\_i on 2, 3

* ex:

proc main() {

assert(N > 0)

assret(1 = X0 ∧ X = X) % implied (VC 1)

Y := 1;

assert(Y = X0 ∧ X = X) % asn

Z := X;

assert(Y = ZN – N ∧ Z = X) % asn

K := N;

assert(Y = ZN – k ∧ Z = X) % asn (loop invariant)

while !(k == 0) do {

assert(Y = ZN – k ∧ Z = X ∧ !(k = 0) % partial-while

assert(Y \* Z = ZN – k + 1 ∧ Z – X) % implied (VC 2)

K := K – 1;

assert(Y \* Z = ZN – k ∧ Z = X) % asn

Y := Y \* Z;

assert(Y = ZN – k ∧ Z = X) % asn

};

assert(Y = ZN – k ∧ Z = X ∧ k = 0) % partial-while

assert(Y = XN) % implied (VC 3)

}

VC 1: self-explanatary

VC 2: Y = ZN – k ∧ Z = X ∧ !(k = 0) |− Y \* Z = ZN – K + 1 ∧ Z = X

1) Y = ZN – k ∧ Z = X ∧ !(k = 0) premise

2) Y = ZN – k by and\_e on 1

3) Z = X by and\_e on 1

4) !(k = 0) by and\_e on 1

5) Y \* Z = ZN – k + 1 by arith % on 2

6) Y \* Z = ZN – k + 1 ∧ Z = X by and\_i on 3, 5

VC 3: Y = ZN – k ∧ Z = X ∧ k = 0 |− Y = XN

1) Y = ZN – k ∧ Z = X ∧ k = 0 premise

2) Y = ZN – k by and\_e on 1

3) Z = X by and\_e on 1

4) k = 0 by and\_e on 1

5) Y = XN – k by eq\_e on 2, 3

6) Y = XN by arith % on 4, 5