Tutorial 3

Adapted from:

Foundations of Programming Languages, Xinyu Feng, Lec 7. Hoare Logic (https://cs.nju.edu.cn/xyfeng/teaching/FOPL/lectureNotes/07_Hoare.pdf)

<<A formally verified NAT>>ACM SIGCOMM, 2017

(https://necto.github.io/online-cv/assets/documents/vignat-slides.pdf)

Program Verification, ETH Zurich, Spring Semester 2019, Lec 8. The Boogie Intermediate Verification Language

UW CSE 507, 19au, Lec 13 Reasoning about Programs II

Proving Partial Correctness

$$\{\textbf{true}\}\ \textbf{while}\ x \neq 10\ \textbf{do skip}\ \{x = 10\}$$

Proof:

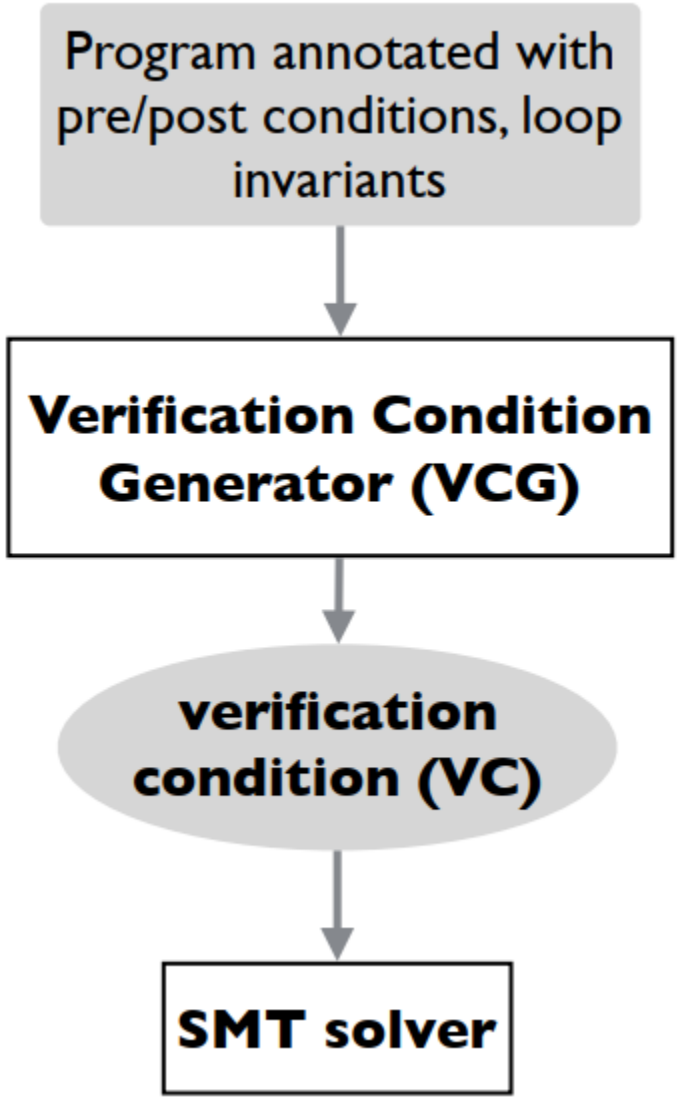
1. $\{\textbf{true} \wedge x \neq 10\}\ \textbf{skip}\ \{\textbf{true} \wedge x \neq 10\}$ SK
2. $\textbf{true} \wedge x \neq 10 \Rightarrow \textbf{true}$
3. $\{\textbf{true} \wedge x \neq 10\}\ \textbf{skip}\ \{\textbf{true}\}$ WC, 1, 2
4. $\{\textbf{true}\}\ \textbf{while}\ x \neq 10\ \textbf{do skip}\ \{\textbf{true} \wedge \neg(x \neq 10)\}$ WHP, 3
5. $\textbf{true} \wedge \neg(x \neq 10) \Rightarrow x = 10$
6. $\{\textbf{true}\}\ \textbf{while}\ x \neq 10\ \textbf{do skip}\ \{x = 10\}$ WC, 4, 5

Testing: Easy but Incomplete

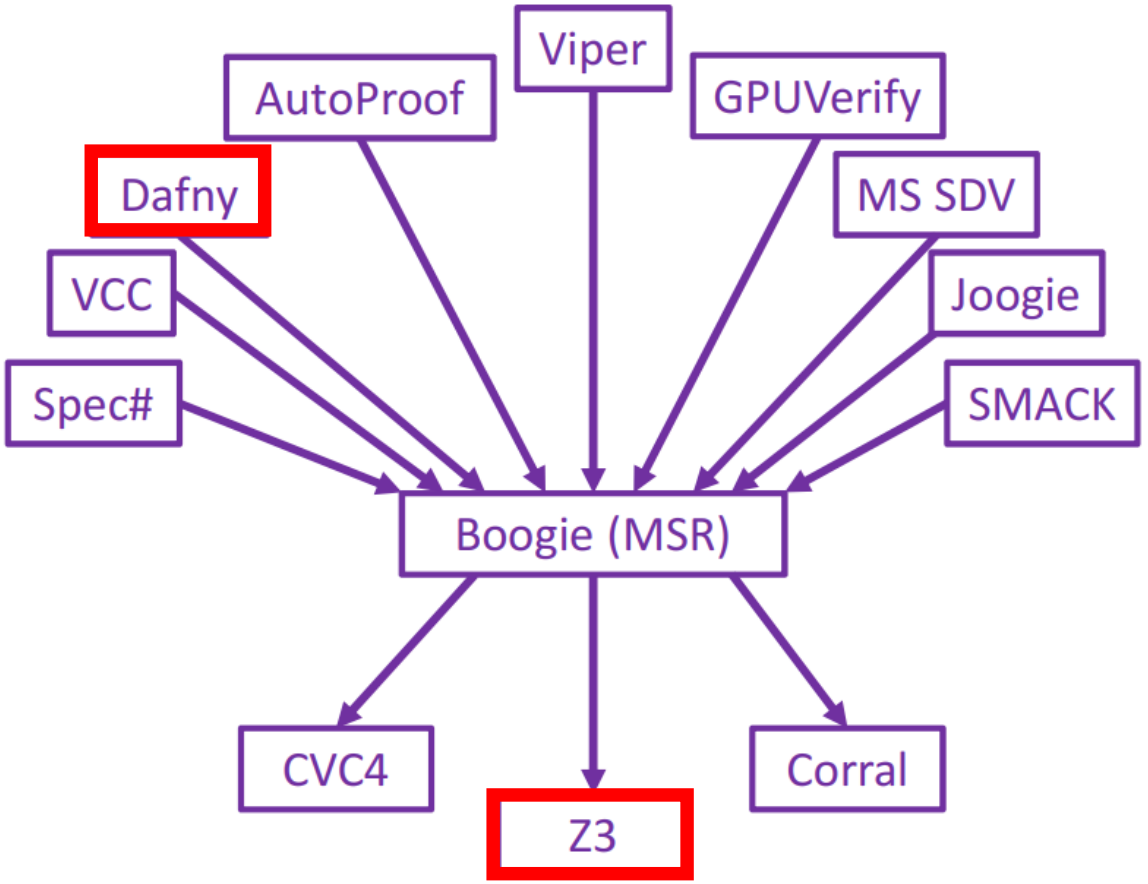
Formal Verification: Complete but Expensive

Formal Verification: Complete ?

Automated Verifiers



Automated Verifiers



Dafny

```
int BinarySearch(int a[], int len, int value)
{
  int low=0, high=len;
  while(low < high) {
    int mid = (low + high) / 2;
    if(a[mid] < value)
      low := mid + 1;
    else if(value < a[mid])
      high := mid;
    else
      return mid;
  }
  return -1;
}
```

二分查找，返回数组中元素的下标

Dafny

```
int BinarySearch(int a[], int len, int value)
{
  int low=0, high=len;
  while(low < high) {
    int mid = (low + high) / 2;
    if(a[mid] < value)
      low := mid + 1;
    else if(value < a[mid])
      high := mid;
    else
      return mid;
  }
  return -1;
}
```

```
predicate sorted(a: array<int>)
   reads a
{
   forall j, k :: 0 <= j < k < a.Length ==> a[j] <= a[k]
}
method BinarySearch(a: array<int>, value: int) returns (index: int)
   requires 0 <= a.Length && sorted(a)
   ensures 0 <= index ==> index < a.Length && a[index] == value
   ensures index < 0 ==> forall k :: 0<=k<a.Length ==> a[k] != value
{
   var low, high := 0, a.Length;
   while low < high
      invariant 0 <= low <= high <= a.Length
      invariant forall i ::
         0<=i<a.Length && !(low <= i < high) ==> a[i] != value
   {
      var mid := (low + high) / 2;
      if a[mid] < value
         {low := mid + 1;}
      else if value < a[mid]
         {high := mid;}
      else
         {return mid;}
   }
   return -1;
}
```

```
Dafny 2.3.0.10506

Dafny program verifier finished with 3 verified, 0 errors
Program compiled successfully
```

Komodo: Using verification to disentangle secure-enclave hardware from software

Andrew Ferraiuolo
Cornell University

Andrew Baumann
Microsoft Research

Chris Hawblitzel
Microsoft Research

Bryan Parno
Carnegie Mellon University

5 SPECIFICATION

We specify and verify Komodo using Dafny [51], a general-purpose verification language. This section describes our trusted Dafny specifications of ARM assembly language (§5.1) and of Komodo's overall correctness (§5.2). We increase our confidence in the Komodo specification by proving several high-level lemmas: that it maintains consistency

Dafny

- Tutorial
  - https://rise4fun.com/Dafny/tutorial
- Answer
  - https://github.com/bor0/dafny-tutorial

```
1 method MultipleReturns(x: int, y: int) returns (more: int, less: int)
2    requires 0 < y
3    ensures less < x < more
4 {
5    more := x + y;
6    less := x - y;
7 }
8
```

'►' shortcut: Alt+B

```
Dafny 2.3.0.10506

Dafny program verifier finished with 1 verified, 0 errors
Program compiled successfully
```

Q & A