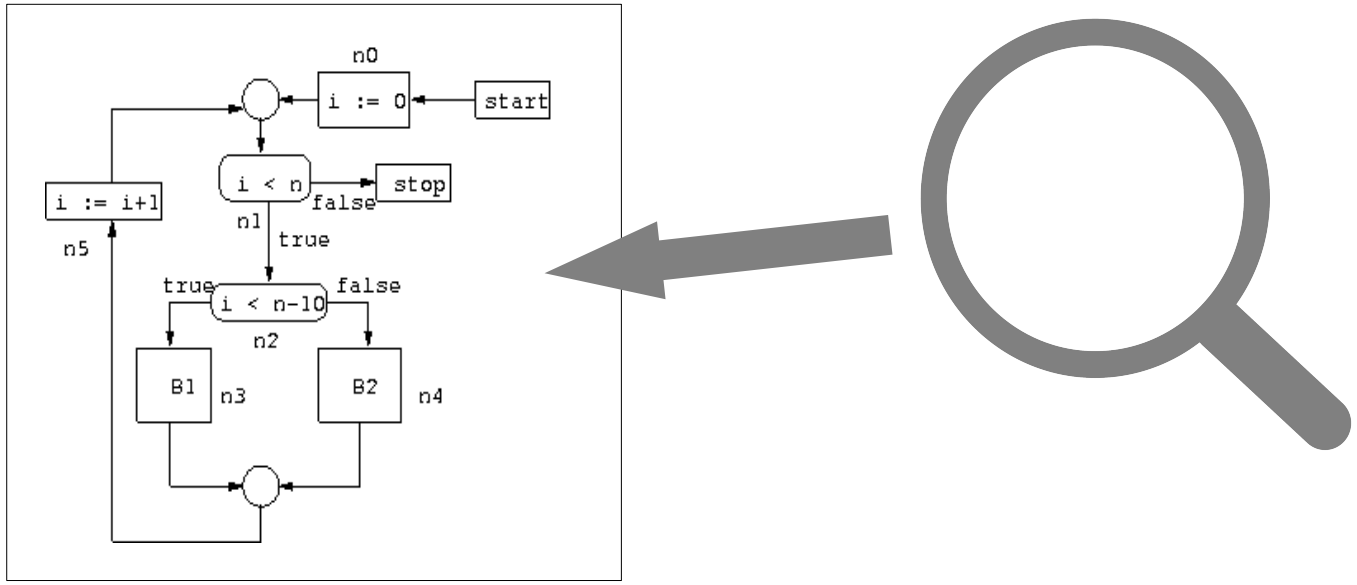
Static Program Analysis
Lecture 3: Different Kinds of Static Analysis



Different Kinds of Static Program Analysis

Syntactically oriented methods:

- Syntactical checking for “bad” patterns indicating bugs
- Type checking

Semantically oriented methods:

- Model checking (often called “formal method”)
- Static Program Analysis (as it is usually known)

(I am extending the meaning of static program analysis as compared with common terminology)

Syntax Checking

Detecting suspicious patterns indicating bugs, and cases of “bad programming practice”

Example (C):

```
int f() { while (b != 0) { if (c = 'x') break; c++; } }
```

Correct C, but some suspicious syntax patterns:

- suspected infinite loop (variables in loop condition are never re-assigned in the loop body)
- no `return` statement from function `f`
- test in `if` is an assignment

Type Checking

Type checking can also be seen as a kind of static program analysis

A type-correct program does not suffer from certain errors (like adding a `char` to a `float`)

In a *statically typed* language (like java) a program must be type-correct in order to compile. The compiler does the type checking

Also in *dynamically typed* (Erlang) or *untyped* languages (C), it's usually good practice to abide to type rules but the compiler won't enforce them

A standalone tool can check such languages for possible type errors. Or, the compiler can issue warnings

Semantics-based Static Program Analysis

Based on the *formal semantics* of the program

The analysis is *fully automated*

Usually works by setting up a set of *equations*, capturing some *property*, which are then solved

Some *approximations* are typically made when they are set up, to make them solvable by automatic means

Originally used in optimizing compilers, but can also be used for program verification

Different Kinds of Semantics-based Analyses

- Dataflow analysis
- Constraint-based analysis
- Abstract interpretation
- Type systems

We will exemplify with a dataflow analysis, and a so-called “value analysis” based on abstract interpretation, in the next lectures