

Static Program Analysis

Summer Semester 2011

Reinhard Wilhelm

with Jörg Herter

<http://rw4.cs.uni-saarland.de/teaching/spa11>

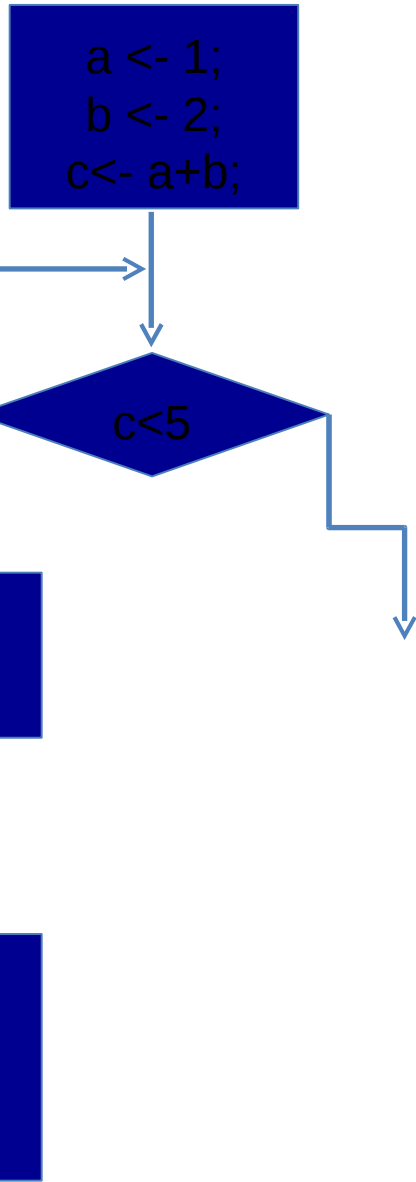
Static Program Analysis

- Origins and Applications -

- Origins:
Compilers – checking the applicability of optimizing program transformations,
examples:
 - moving computations from run time to compile time
 - avoiding redundant computations
- Hot application area:
Verification – proving safety properties
examples:
 - proving the absence of run-time errors

Literature used in the course

- F. Nielson, H. Riis Nielson, C. Hankin: Principles of program analysis. Springer 2005: I-XXI, 1-452
- H. Seidl, R. Wilhelm, S. Hack: Übersetzerbau – Analyse und Transformation, Springer, 2010
- H. Seidl, R. Wilhelm, S. Hack: Compiler Design – Program Analysis and Transformation, Springer, to appear



Problem:

Determine at each program point the values that program variables have every time control reaches this point.

Purpose:

Folding away (sub-)expressions whose operands are statically known, evaluate them at compile time instead of at run time.

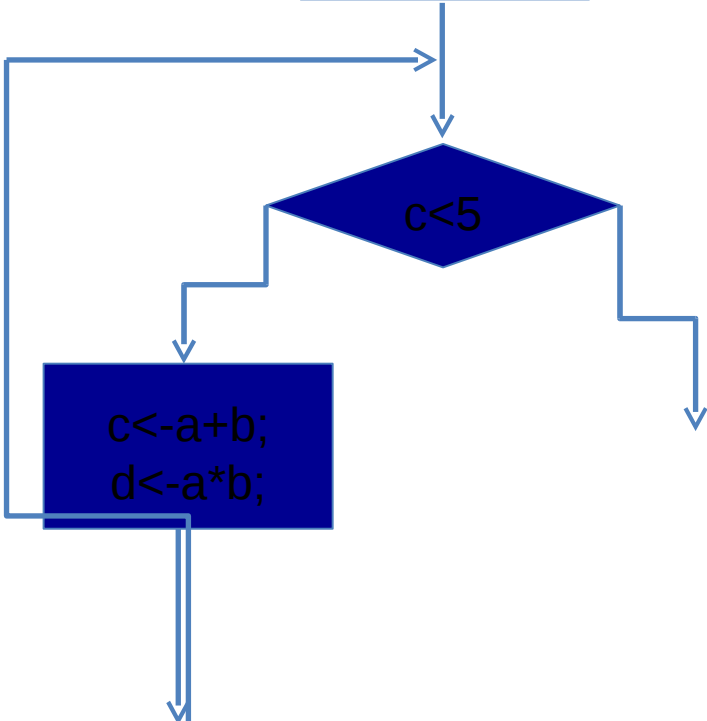
The necessary ingredients:

```
a <- 1;  
b <- 2;  
c <- a+b;
```

$c < 5$

```
c <- a+b;  
d <- a*b;
```

```
d <- c-1;  
a <- 2;  
b <- -1;  
c <- a+b;
```



```
a <- 1;
b <- 2;
c <- a+b;
```

Disadvantage of this representation:
Need to associate information with positions before and after statements.

c < 5

```
c <- a+b;
d <- a*b;
```

```
d <- c-1;
a <- 2;
b <- 1;
c <- a+b;
```

Alternative:

```
a <- 1;
b <- 2;
c <- a+b;
```

c < 5

c >= 5

```
c <- a+b;
d <- a*b;
```

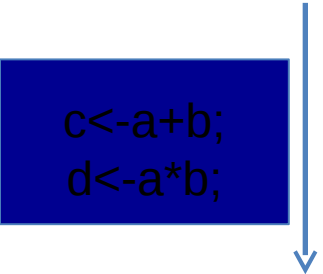
Statements at edges

```
d <- c-1;
a <- 2;
b <- 1;
c <- a+b;
```

Information at nodes

Outgoing edges at conditionals labeled with condition and negation of condition

Program Analysis should be Semantics-based



```
c<-a+b;  
d<-a*b;
```

statements labeling edges
have a (concrete) semantics,
called *concrete edge effects*.

Alternative specifications:
NNH: equalities
SWH: inequalities

They typically transform the state
before the execution of the statement
to the state after the execution of the statement

For a semantics-based static program analysis
edges also have an abstract semantics,
called *abstract edge effects*.

In the case of constant folding,
they transform abstract variable bindings
into abstract variable bindings.

