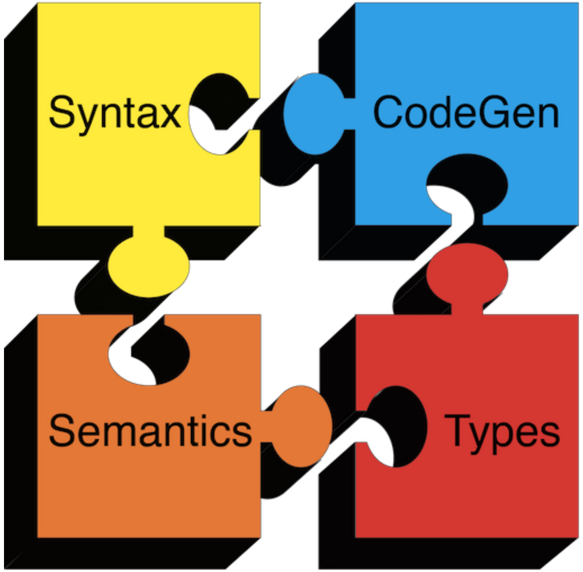


Compilers

Dataflow Analysis

Alex Aiken

Dataflow Analysis

Recall the simple basic-block optimizations

- Constant propagation
- Dead code elimination

```
X := 3
Y := Z * W
Q := X + Y
```

→

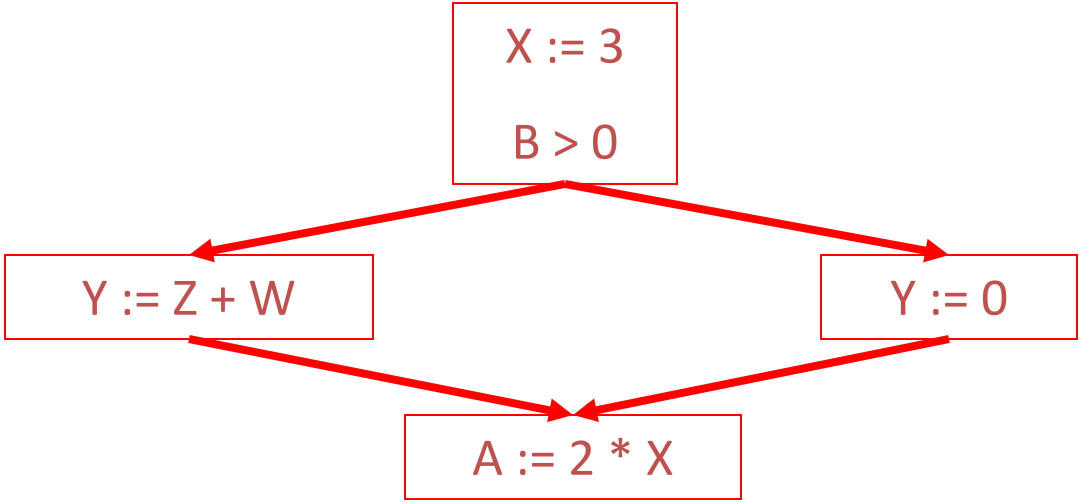
```
X := 3
Y := Z * W
Q := 3 + Y
```

→

```
Y := Z * W
Q := 3 + Y
```

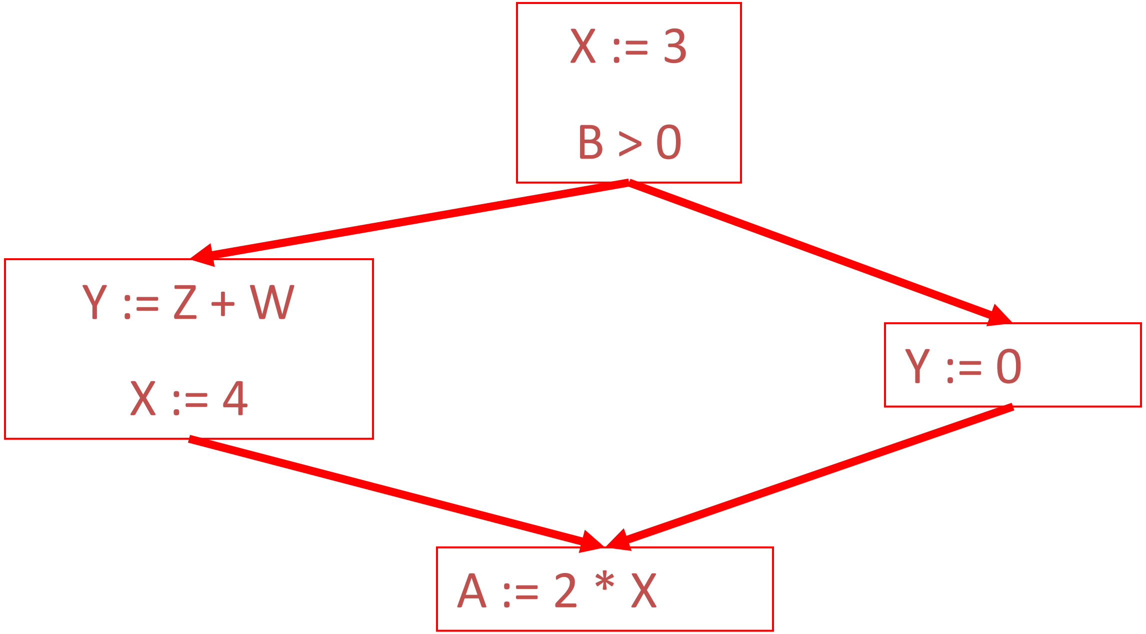
Dataflow Analysis

These optimizations can be extended to an entire control-flow graph



Dataflow Analysis

- How do we know it is OK to globally propagate constants?

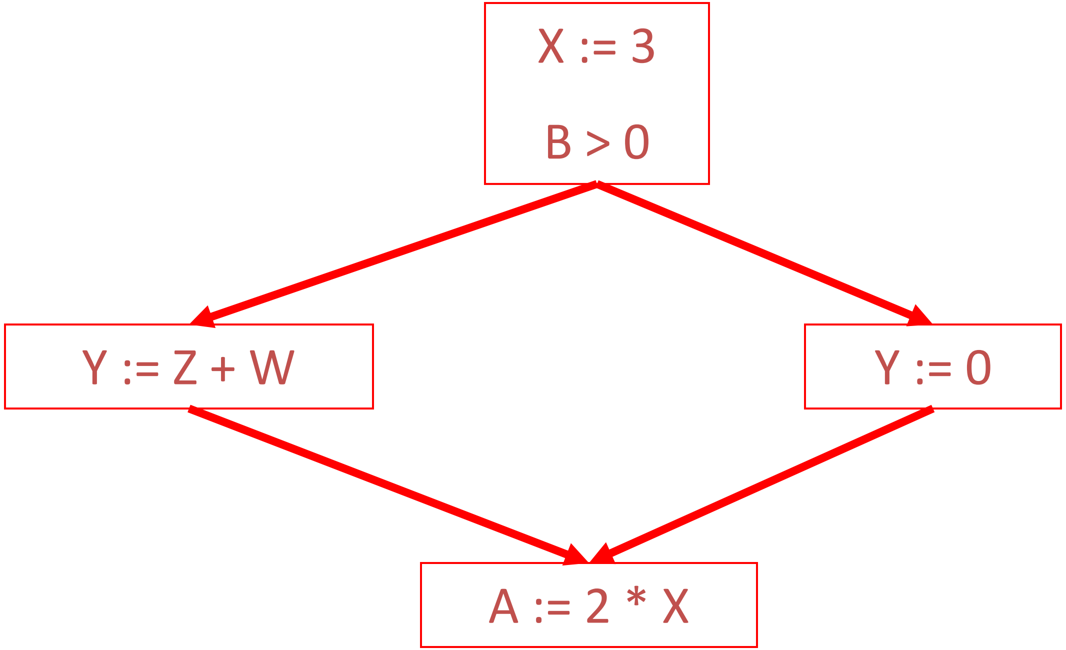


Dataflow Analysis

To replace a use of x by a constant k we must know:

On every path to the use of x, the last assignment to x is
x := k

Dataflow Analysis



X := 3

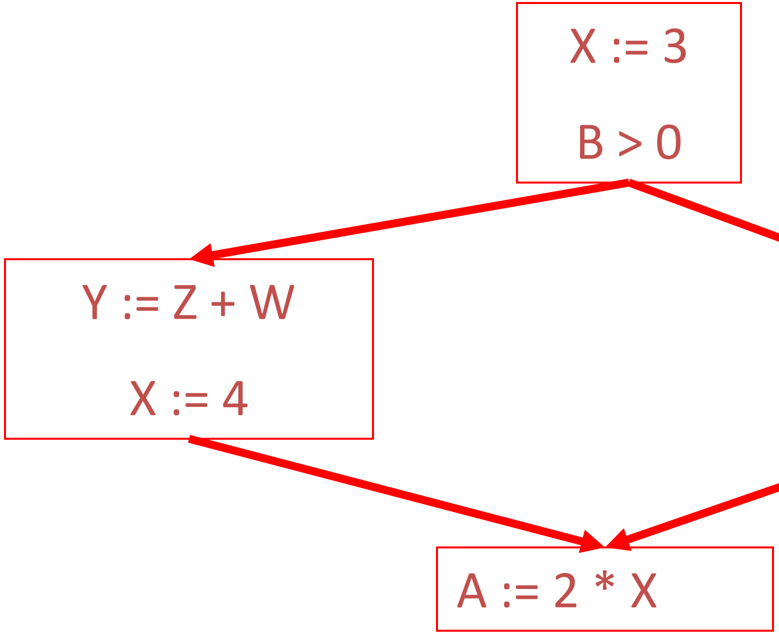
B > 0

Y := Z + W

Y := 0

A := 2 * X

Dataflow Analysis



X := 3

B > 0

Y := Z + W

X := 4

Y := 0

A := 2 * X

Dataflow Analysis

- The correctness condition is not trivial to check
- “All paths” includes paths around loops and through branches of conditionals
- Checking the condition requires *global dataflow analysis*
  - An analysis of the entire control-flow graph

Dataflow Analysis

Global optimization tasks share several traits:

- The optimization depends on knowing a property X at a particular point in program execution
- Proving X at any point requires knowledge of the entire program
- It is OK to be conservative. If the optimization requires X to be true, then want to know either
  - X is definitely true
  - Don't know if X is true
  - It is always safe to say "don't know"

Dataflow Analysis

- *Global dataflow analysis* is a standard technique for solving problems with these characteristics

- Global constant propagation is one example of an optimization that requires global dataflow analysis