Dataflow Analysis

Recall the simple basic-block optimizations

- Constant propagation
- Dead code elimination

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

to

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

to (if X is not used anywhere else)

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

These optimizations can be extended to an entire control-flow graph (CFG).

```
BB1:
X := 3
B > 0 then goto BB2
         else goto BB3

BB2:
Y := Z + W

BB3:
Y := 0

BB4:
A := 2 * X
```

Here we can substitute X with 3 in BB4.

Another example

```
BB1:
X := 3
B > 0 then goto BB2
         else goto BB3

BB2:
Y := Z + W
X := 4

BB3:
Y := 0

BB4:
A := 2 * X
```

Now we cannot replace X with 3 in BB4 even though X is only assigned constants.

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

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`.*

Let's look at the first example. Indeed this condition is satisfied for X:=3 at BB4.

Let's look at the second example. This condition is *not* satisfied for `X` at BB4.

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*, i.e., an analysis of the entire CFG.

In general, 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".

Global dataflow analysis is a standard technique for solving problems with these characteristics. Global constant propagation is an example of such a problem.