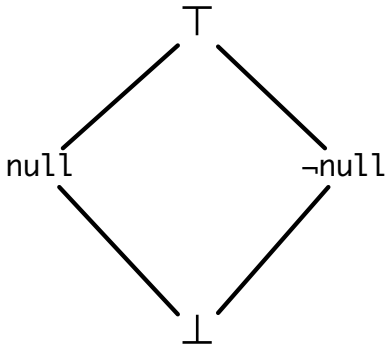
ECE 573
Problem Set 9: Dataflow analysis

In this problem set, your goal is to develop a dataflow analysis to find `null` values. A common error in programs is dereferencing `null` values, so a dataflow analysis which can detect such dereferences can be a valuable debugging tool (imagine a compiler that flags potential null dereferences for you, before you run the program). What you should build is a dataflow analysis that determines, at each step, whether a variable is *definitely not null*, *definitely null* or *may be null*.

A hint to keep in mind as you develop this analysis: think about how this analysis relates to the constant propagation analysis).

1. What is the lattice that you should use for this analysis?



   In this lattice, $\bot$ represents "no information", `null` means "definitely null," `¬null` means "definitely not null," and $\top$ means "maybe null." Note the similarity of this lattice to the constant propagation lattice (we can think of "maybe null" as being the same thing as "definitely not constant").

2. Which direction should this analysis use?

   This should be a forward analysis, just like constant propagation. At each step, the dataflow information depends on what has happened in the past.

3. What is the confluence operator?

   Join ($\sqcup$). This makes intuitive sense: if you are at a merge point and along one incoming branch you know a variable is null, and along the other you know it is not, then after the merge the variable *may be* null.

4. Give the transfer functions for the following statements (for each transfer function, specify which variables will *change* their states, and under which conditions).

- `x := null`

  $x_{out} =$ `null`
- `if (x == null) then { ... } else { ... }`

  $x_{true} =$ `null` and $x_{false} = \neg$`null`

  You can be extra clever here, and not propagate information along one of the paths at all, if you know the value of x_{in} (like we did in class for constant propagation), but I will not require that.
- `x := y`

  $x_{out} = y_{in}$
- `x := &z`

  $x_{out} = \neg$`null`

5. Argue that the transfer functions you developed in step (4) are monotonic.

   3 of the 4 transfer functions are constant functions (the output is independent of the input), so they are trivially monotonic. The only interesting transfer function is for `x := y`. In this case, we see that if y_{in} increases, x_{out} increases, so this function is also monotonic.

6. How should this analysis be initialized?

   Just like constant propagation. The variables at every program point should be initialized to $\bot$, except for at the beginning of the program, where the variables should be initialized to $\top$.

7. Show the results of your dataflow analysis for the following piece of code:

   This is more or less straightforward. Note that because of the conditionals, `x` is $\neg$`null` after `x = y`.

```
   : x := 4;
   : y := null;
L1 : if (y == null) goto L2;
   :    x = y;
   :    if (x == null) goto L3;
   :       x = 5;
   :       goto L4;
L3 :       x = null;
L4 :    y = x;
   :    goto L1;
L2 : end;
```