A Refined Example

Consider the following program:

```c
int x, y;
if (cond) {
    x = 1;
} else {
    x = 2;
}
y = x - x;
```

Step 1: Define the Environments

- On the **true branch**, we have an environment $d_1 = \{x = 1\}$.
- On the **false branch**, we have an environment $d_2 = \{x = 2\}$.

Step 2: Apply the Transfer Function Separately

The transfer function f corresponds to the assignment $y = x - x$.

- **For d_1:**
  With $x = 1$, we compute:

  $$y = 1 - 1 = 0.$$

  So, $f(d_1) = \{y = 0\}$.

- **For d_2:**
  With $x = 2$, we compute:

  $$y = 2 - 2 = 0.$$

  So, $f(d_2) = \{y = 0\}$.

Now, taking the join (or meet, depending on the lattice convention) of the results:

$$f(d_1) \sqcup f(d_2) = \{y = 0\}.$$

Step 3: Join the Environments First, Then Apply the Transfer Function

First, join the environments d_1 and d_2:

$$d_1 \sqcup d_2 = \{x = 1 \sqcup 2\} = \{x = \top\},$$

since 1 and 2 are different, and the join in constant propagation yields $\top$ (unknown).

Now, apply the transfer function f to the joined environment:

- With $x = \top$, evaluating $y = x - x$ yields:

  $$y = \top - \top = \top,$$

  because the subtraction of unknowns is unknown.

So,

$$f(d_1 \sqcup d_2) = \{y = \top\}.$$

Step 4: Compare the Two Results

- **Right-hand side (applying f first then joining):**
  $f(d_1) \sqcup f(d_2) = \{y = 0\}.$
- **Left-hand side (joining first then applying f):**
  $f(d_1 \sqcup d_2) = \{y = \top\}.$

Since $0 \neq \top$, we see that:

$$f(d_1 \sqcup d_2) \neq f(d_1) \sqcup f(d_2).$$

What This Means

The fact that the two orders of operations yield different results is the hallmark of **non-distributivity**. The reason constant propagation is non-distributive is that **information can be lost when merging environments** before applying a transfer function. In our refined example, joining the two different constants for x results in an unknown value, which then leads to an unknown result for y. However, if you apply the transfer function before joining, you obtain a precise value (0 in both cases) and then join these precise results.

Thus, although the simple $y = x + 1$ example did not exhibit a difference (both orders ended up with $y = \top$), the $y = x - x$ example clearly shows that constant propagation does not distribute over the join operation in the IFDS framework.

Does this help clarify why constant propagation is considered non-distributive?