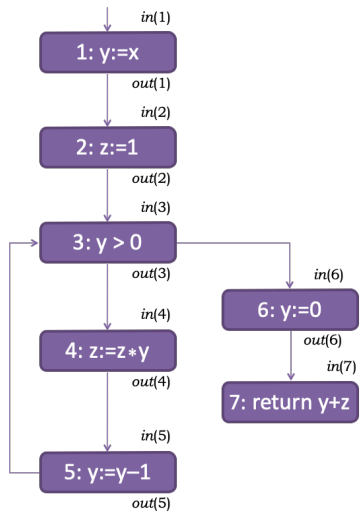
Section 1. Transition Functions

Static analysis is typically done on some intermediate representation of the program. One such convenient representation is the program's *control-flow graph* (CFG). The CFG comprises of basic blocks, and directed edges connecting blocks with possible flow of execution between them. In order to connect the semantics of the program to the analysis domain (which we defined, recall, as a *lattice*), we need to define a family of functions that describe, for every possible block ℓ, a mapping from $in(\ell)$ to $out(\ell)$, according to how we perceive the update of "known facts" when program state changes by executing each block. These are called transfer functions, and denoted $f_\ell : \mathrm{L} \to \mathrm{L}$, where L is the domain lattice.



A basic block may contain any number of statements or instructions, either in the source language or some IR (such as 3AC). In most of our examples, it is convenient to assume that each block corresponds to a single statement from the source language (which, of course, may translate to multiple IR instructions). We then have to describe the transfer function f_ℓ for every possible statement that may occur in the block ℓ.

For example, in the lecture we saw the Reaching Definition analysis, in which the transfer functions were given as (slide 39):

Statement in ℓ	Transfer function f_ℓ
x := *expr*	$f_\ell(v) = v \setminus \{(x,\ell') \mid \ell' \in \mathrm{Lab}\} \cup \{(x, \ell)\}$
any other statement	$f_\ell(v) = v$

Section 2. Flow equations

The *Monotone Framework* prescribes the construction of *flow equations* (as shown in slide 59):

$$in(\ell) = \begin{cases} \text{Initial} & \text{if } \ell \in \text{Entry labels} \\ \bigsqcup \{ \text{out}(\ell') \mid (\ell',\ell) \in \text{CFG edges} \} & \text{otherwise} \end{cases}$$

$$out(\ell) = T_\ell(in(\ell))$$

And these equations, in turn, induce a compound function, that we marked as F, that characterizes the system of flow equations in the sense that a vector of values $\overline{v}$ is a a solution to the flow equations if and only if it is a *fixed point* of F, that is, $F(\overline{v}) = \overline{v}$.

Why is that good? Because from the Knaster-Tarski theorem we know that if F is *monotonic* — that is, $\forall x, y.\ x \sqsubseteq y \Rightarrow F(x) \sqsubseteq F(y)$ — and if F has a fixed point, then it has a *least fixed point*. Furthermore, Kleene's theorem suggests an algorithm for computing this least fixed point. This least fixed point is of interest to us because it represents the **most precise solution** to the flow

equations. Recall that the flow equations describe the data-flow semantics of our program; therefore the most precise solution to them will give us the **most information** about the program.

Now, in order to utilize these nice properties from the Knaster-Tarski and Kleene theorems, we need to prove the monotonicity of F. Notice that F operates on a product lattice L^n, where n is the number of $in(\ell)$ and $out(\ell)$ "unknowns" that occur in the flow equations. In a product lattice, the partial order relation $\sqsubseteq$ is defined *pointwise* (slide 31). As a consequence, it is sufficient to prove that each of the f_ℓ s comprising F is monotonic. The join operation, $\sqcup$, is always monotonic by definition (you are invited to prove this as an exercise).

Let us show that for Reaching Definitions, the transfer functions are monotonic. The domain lattice is a power set lattice, $\mathrm{L} = \mathscr{P}(\text{Var} \times \text{Lab})$ with order $\sqsubseteq\ = \subseteq$.

If ℓ is x := *expr*, then let $v_1, v_2 \in \mathrm{L}$ such that $v_1 \sqsubseteq v_2$. That is, $v_1 \subseteq v_2$. Therefore:

$$f_\ell(v_1) \;=\; v_1 \backslash \{(\mathrm{x}, \ell') \mid \ell' \in \text{Lab}\} \cup \{(x, \ell)\} \quad \subseteq \quad v_2 \backslash \{(\mathrm{x}, \ell') \mid \ell' \in \text{Lab}\} \cup \{(x, \ell)\} \;=\; f_\ell(v_2)$$

This proposition proved to be quite trivial, because we can rely on the monotonicity of the set operations \ and ∪, and since the set operands $\{(x, \ell) \mid \ell' \in \text{Lab}\}$ and $\{(x, \ell)\}$ **are the same** in $f_\ell(v_1)$ as in $f_\ell(v_2)$. In fact, all transfer functions that take the form $f_\ell(v) = v \setminus K \cup G$ are monotonic for the exact same reason — **as long as K and G do not depend on v**.

The second case is when is ℓ not an assignment statement. In this case f_ℓ is the identity function, which is trivially monotonic. This concludes the monotonicity proof of all of the transfer functions f_ℓ for Reaching Definitions, and, consequently, of the function F that is constructed for the fixed-point computation.

Since the lattice $\mathrm{L} = \mathscr{P}(\text{Var} \times \text{Lab})$ has *no infinite chains*, it follows that data-flow analysis using the monotone framework *converges*, that is, will compute the least fixed point in finite time.