Data Flow Analyses: Principles
Intraprocedural Analysis

Classical analyses:

- Available Expressions Analysis
- Reaching Definitions Analysis
- Very Busy Expressions Analysis
- Live Variables Analysis
Components of Data Flow Analysis

- **Property Spaces:** how do we formally represent the set of data flow facts that we want to compute?
- **Transfer Functions:** how do we formally represent the effect of each program statement on the data flow facts?
- **Instances:** is this a forward or a backward analysis? what are the initial values?
Property Spaces
A (meet) semi-lattice $\mathbf{L} = (S, \sqcap)$ is a set $S$ with an operation, called meet ($\sqcap$), that has the following properties:

(1) For all $x, y \in S$, there exist a unique $z \in S$ such that $x \sqcap y = z$ (CLOSURE).

(2) For all $x, y, z \in S$, we have $x \sqcap x = x$, $x \sqcap y = y \sqcap x$, $x \sqcap (y \sqcap z) = (x \sqcap y) \sqcap z$. (idempotence, commutativity, and associativity).
The meet operator defines *an order relation* \((\sqsubseteq)\):

\[ x \sqsubseteq y \text{ if and only if } x \sqcap y = x \]

For all \( x, y, z \in \mathbb{L} \):

\[ x \sqsubseteq x \text{ (reflexivity).} \]

If \( x \sqsubseteq y \) and \( y \sqsubseteq z \), then \( x \sqsubseteq z \) (transitivity).

If \( x \sqsubseteq y \) and \( y \sqsubseteq x \), then \( x = y \) (antisymmetry).

The unit for the meet operator is \( \top \) (*top)*:

\[ \forall x, \ x \sqcap \top = x \]

It is easy to show that \( \top \) (if it exists) is the greatest element in the semi-lattice, and the semi-lattice is called *complete*. 
A (join) **semi-lattice** \( L = (S, \sqcup) \) is a set \( S \) with an operation, called **join** \((\sqcup)\), that has the following properties:

1. For all \( x, y \in S \), there exist a **unique** \( z \in S \) such that \( x \sqcup y = z \) (CLOSURE).

2. For all \( x, y, z \in S \), we have \( x \sqcup x = x \), \( x \sqcup y = y \sqcup x \), \( x \sqcup (y \sqcup z) = (x \sqcup y) \sqcup z \). (idempotence, commutativity, and associativity).
The meet operator defines *an order relation* ($\sqsubseteq$):

$$ x \sqsubseteq y \text{ if and only if } x \sqcup y = y $$

For all $x, y, z \in \mathbb{L}$:

$$ x \sqsubseteq x \text{ (reflexivity).} $$

If $x \sqsubseteq y$ and $y \sqsubseteq z$, then $x \sqsubseteq z$ (transitivity).

If $x \sqsubseteq y$ and $y \sqsubseteq x$, then $x = y$ (antisymmetry).

The unit for the join operator is $\bot$ (*bottom*):

$$ \forall x, x \sqcup \bot = x $$

It is easy to show that $\bot$ (if it exists) is the least element in the semi-lattice, and the semi-lattice is called *complete*. 
Lattices

A lattice $L = (S, \sqcap, \sqcup)$ is a set $S$ with two operations, a meet ($\sqcap$) and a join ($\sqcup$), such that:

1. $(S, \sqcap)$ is a meet semi-lattice.
2. $(S, \sqcup)$ is a join semi-lattice.
3. Operators $\sqcap$ and $\sqcup$ are linked by corresponding absorption laws:

$$x \sqcup (x \sqcap y) = x \sqcap (x \sqcup y) = x$$

If both $\top$ and $\bot$ exist, then the lattice is called complete.
If we have a set $S$, then subset lattice for $S$ is $(\mathcal{P}(S), \cap, \cup)$.

$\bot = \emptyset$  \hspace{1cm} $\top = S$  \hspace{1cm} $\subseteq \subseteq$
Descending Chains

A *descending chain* is a sequence of elements related by the order relation:

\[ x_1 \sqsupseteq x_2 \sqsupseteq \cdots \sqsupseteq x_n \]

The *height* of a (semi-) lattice is the length of the largest descending chain.

**important property:** finite descending chains.
We are interested in the set of data flow facts \( \mathbb{F} = \text{Var} \times \text{Lab} \).

The reaching definition lattice is \( \mathbb{L}_{RD} = (\mathcal{P}(\mathbb{F}), \cup, \subseteq) \).
We are interested in the set of data flow facts \( \mathbb{F} = \mathbb{AExp} \).

The available expressions lattice is \( \mathbb{L}_{\mathbb{AE}} = (\mathcal{P}(\mathbb{F}), \cap, \supseteq) \).
But, not all lattices are always as simple as the subset lattice ...
The aim of the *Constant Propagation Analysis* is to determine

For each program point, whether or not a variable has a constant value whenever execution reaches that point.

**Example:**

\[
\begin{align*}
[x := 6]^1; [y := 3]^2; & \text{while } [x > y]^3 \text{ do } ([x := x - 1]^4; [z := y \times y]^6)
\end{align*}
\]

The analysis enables a transformation into

\[
\begin{align*}
[x := 6]^1; [y := 3]^2; & \text{while } [x > 3]^3 \text{ do } ([x := x - 1]^4; [z := 9]^6)
\end{align*}
\]
Let $\mathbb{Z} = \mathbb{Z} \cup \{T\}$.

Where $T$ represent a non-constant value.

But, why $T$?

We are in the process of defining join semi-lattice. constant $\sqcup$ non-constant should be non-constant.

This makes non-constant the greater than all constant values.
To be more exact, we define $\sqcup$ over values as:

$$v_1 \sqcup v_2 = \begin{cases} 
  v_1 & \text{if } v_1 = v_2 \\
  \top & \text{if } v_1 \neq v_2
\end{cases}$$
But \( \mathbb{Z} \) is not exactly the set of dataflow facts that we want for this analysis.

The set of data flow facts: \( \mathcal{F} = \sigma : \text{Var} \rightarrow \mathbb{Z} \).

Are we there yet? Not exactly!

We want our join semi-lattice to be complete. This means that we need a bottom element.

Therefore, we add a \( \bot \) value (which corresponds to no information) and a \( \hat{\bot} \) function to go with it.
Let $F = F \cup \{\top\}$.

The constant propagation lattice: $L_{CP} = (F, \sqcup, \sqsubseteq)$.

Note: we could have treated this as a meet semi-lattice, and then, non-constant would become the $\bot$ in that semi-lattice, and $\top$ would stand for no information, which we had to add.
Transfer Functions
We can lift $\sqcup$ to functions $\sigma = \sigma_1 \sqcup \sigma_2$ as:

$$\sigma(x) = \sigma_1(x) \sqcup \sigma_2(x).$$

Which implies:

$$\forall \sigma_1, \sigma_2 \in F :$$

$$\sigma_1 \sqsubseteq \sigma_2 \iff \forall x \in \text{Var}, \ \sigma_1(x) \sqsubseteq \sigma_2(x)$$

And, $\sigma = \sigma \sqcup \bot$, for all $\sigma$, which means:

$$\forall \sigma \in F : \bot \sqsubseteq \sigma.$$
A transfer function models, for a particular data flow analysis problem, the effect of the programming language constructs as a mapping from the lattice (used in the analysis) to itself.

\[ f : L \longrightarrow L \]

Example:

\[ f_{st}^{RD} : \mathcal{P}(Var \times Lab) \longrightarrow \mathcal{P}(Var \times Lab) \]

\[ \forall S \subseteq Var \times Lab : f_{st}^{RD}(S) = S - kill_{RD}(st) \cup gen_{RD}(st) \]
Properties of Transfer Functions

Monotonicity:

\[ \forall x, y \in \mathbb{L} : x \sqsubseteq y \implies f(x) \sqsubseteq f(y). \]

Distributivity:

\[ \forall x, y \in \mathbb{L} : f(x \sqcap y) = f(x) \sqcap f(y). \]

What about the transfer functions that you have already seen?