Lecture 3 – Dataflow Analysis

SWS128: Software Analysis

Jihyeok Park

PLRG

2026 Fall

Recall

- A **CFG**: program points, and edges $p \xrightarrow{c} q$ carrying one command each.
- **Dominators**: a recursive definition, solved by iterating to a **fixpoint** with a worklist.

Recall

- A **CFG**: program points, and edges $p \xrightarrow{c} q$ carrying one command each.
- **Dominators**: a recursive definition, solved by iterating to a **fixpoint** with a worklist.
- Today: the same iteration, but the sets hold **data**: definitions, variables, expressions. Four classic analyses.

Contents

Contents

Dataflow Analysis

A **dataflow analysis** asks: which data flows where, and how.

Dataflow Analysis

A **dataflow analysis** asks: which data flows where, and how.

To define one, fix:

- **Data** $\mathbb{D}$: the facts tracked at each point, and a **join** to merge them

$$\Gamma : N \to \mathbb{D} \qquad\qquad \sqcup : \mathbb{D} \times \mathbb{D} \to \mathbb{D}$$

($\sqcup$ is $\cup$ for **may**: some path, or $\cap$ for **must**: every path)

Dataflow Analysis

A **dataflow analysis** asks: which data flows where, and how.

To define one, fix:

- **Data** $\mathbb{D}$: the facts tracked at each point, and a **join** to merge them

$$\Gamma : N \to \mathbb{D} \qquad\qquad \sqcup : \mathbb{D} \times \mathbb{D} \to \mathbb{D}$$

  ($\sqcup$ is $\cup$ for **may**: some path, or $\cap$ for **must**: every path)

- **Transfer**: what one edge $\ell = p \xrightarrow{c} q$ does to the data

$$f_\ell(X) = \mathsf{gen}(\ell) \cup (X \setminus \mathsf{kill}(\ell))$$

  - kill(ℓ): the facts ℓ destroys
  - gen(ℓ): the facts ℓ creates

Dataflow Analysis

A **dataflow analysis** asks: which data flows where, and how.

To define one, fix:

- **Data** $\mathbb{D}$: the facts tracked at each point, and a **join** to merge them
  $$\Gamma : N \to \mathbb{D} \qquad\qquad \sqcup : \mathbb{D} \times \mathbb{D} \to \mathbb{D}$$
  ($\sqcup$ is $\cup$ for **may**: some path, or $\cap$ for **must**: every path)
- **Transfer**: what one edge $\ell = p \xrightarrow{c} q$ does to the data
  $$f_\ell(X) = \mathsf{gen}(\ell) \cup (X \setminus \mathsf{kill}(\ell))$$
  - $\mathsf{kill}(\ell)$: the facts ℓ destroys
  - $\mathsf{gen}(\ell)$: the facts ℓ creates
- **Direction**: how one round moves the data across the edges
  $$\begin{array}{lll} \text{forward} & F(\Gamma)(q) = \bigsqcup_\ell f_\ell(\Gamma(p)) & \text{where } \ell = p \xrightarrow{c} q \\ \text{backward} & F(\Gamma)(p) = \bigsqcup_\ell f_\ell(\Gamma(q)) & \text{where } \ell = p \xrightarrow{c} q \end{array}$$

Dataflow Analysis

A **dataflow analysis** asks: which data flows where, and how.

To define one, fix:

- **Data** $\mathbb{D}$: the facts tracked at each point, and a **join** to merge them

$$\Gamma : N \to \mathbb{D} \qquad\qquad \sqcup : \mathbb{D} \times \mathbb{D} \to \mathbb{D}$$

($\sqcup$ is $\cup$ for **may**: some path, or $\cap$ for **must**: every path)

- **Transfer**: what one edge $\ell = p \xrightarrow{c} q$ does to the data

$$f_\ell(X) = \mathsf{gen}(\ell) \cup (X \setminus \mathsf{kill}(\ell))$$

  - $\mathsf{kill}(\ell)$: the facts ℓ destroys
  - $\mathsf{gen}(\ell)$: the facts ℓ creates

- **Direction**: how one round moves the data across the edges

$$\begin{array}{lll} \text{forward} & F(\Gamma)(q) = \bigsqcup_\ell f_\ell(\Gamma(p)) & \text{where } \ell = p \xrightarrow{c} q \\ \text{backward} & F(\Gamma)(p) = \bigsqcup_\ell f_\ell(\Gamma(q)) & \text{where } \ell = p \xrightarrow{c} q \end{array}$$

- **Initial** Γ_0: the start point keeps it; elsewhere iterate $\Gamma_{i+1} = F(\Gamma_i)$ to a fixpoint, with the worklist.

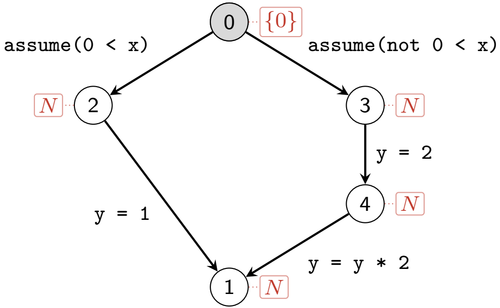
Dominators as a Dataflow Analysis

Which points must have run before p?

- **Data**: $\mathbb{D} = \mathcal{P}(N)$
- **Join**: $\cap$ (**must**: true on every path)
- **Transfer**: $\mathsf{kill}(\ell) = \emptyset$, $\mathsf{gen}(\ell) = \{q\}$ where $\ell = p \xrightarrow{c} q$
- **Direction**: forward
- **Initial**: $\Gamma_0(\texttt{entry}) = \{\texttt{entry}\}$, $\Gamma_0(p) = N$ otherwise

$$F(\Gamma)(q) = \bigcap_{\ell}(\Gamma(p) \cup \{q\}) = \{q\} \cup \bigcap_{p \to q} \Gamma(p)$$

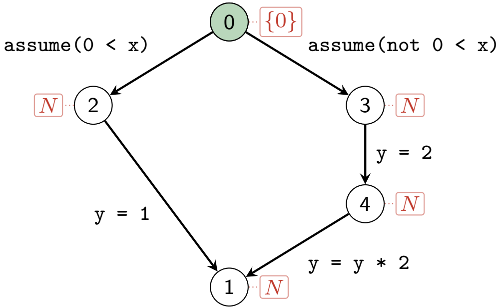
Dominators, Step by Step



Data $\mathbb{D} = \mathcal{P}(N)$

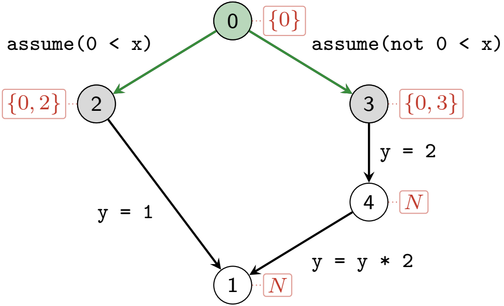
Initial $\Gamma_0(\texttt{entry}) = \{\texttt{entry}\}$, $\Gamma_0(p) = N$ otherwise

Dominators, Step by Step



Direction forward: $W = \{\texttt{entry}\}$; pop p, push along each $\ell = p \xrightarrow{c} q$

Dominators, Step by Step

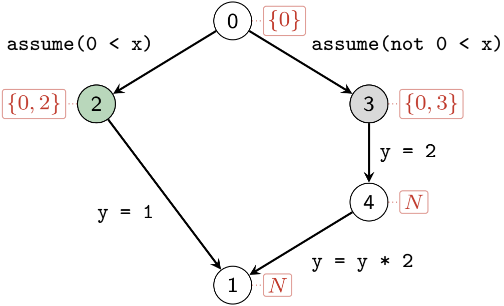


$$\textbf{Transfer}\ \text{kill}(\ell) = \emptyset,\ \text{gen}(\ell) = \{q\} \quad \text{where } \ell = p \xrightarrow{c} q$$

$$\textbf{Join}\ \sqcup = \cap$$

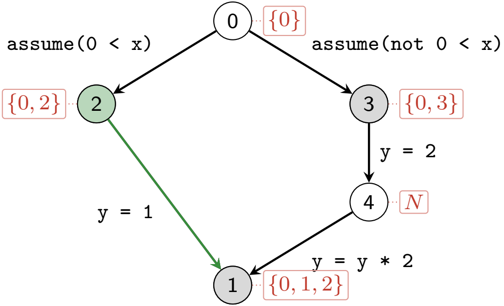
$\Gamma(2)$, $\Gamma(3)$ changed: Push 2, 3 to W

Dominators, Step by Step



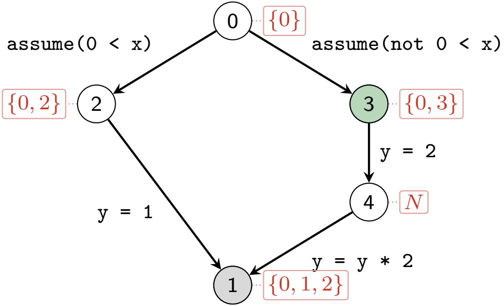
Pop 2 from $W = \{2, 3\}$

Dominators, Step by Step



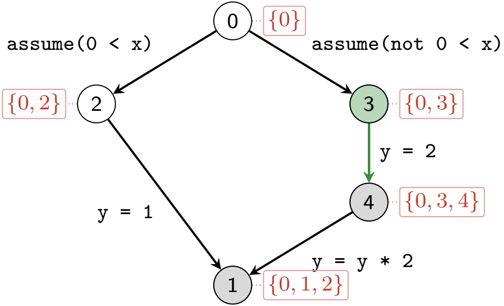
$\Gamma(1) = N \cap \{0, 2, 1\} = \{0, 1, 2\}$ Push 1 to W

Dominators, Step by Step



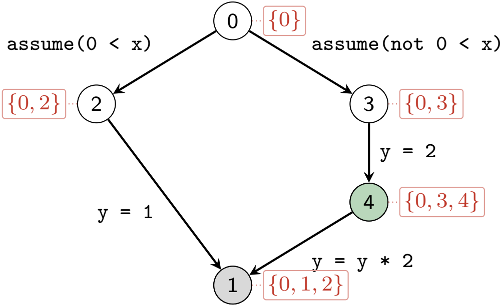
Pop 3 from $W = \{3, 1\}$

Dominators, Step by Step



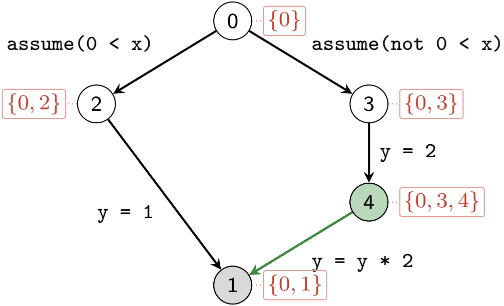
$\Gamma(4) = N \cap \{0, 3, 4\} = \{0, 3, 4\}$ Push 4 to W

Dominators, Step by Step



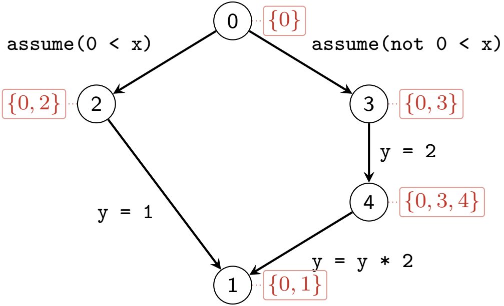
Pop 4 from $W = \{1, 4\}$

Dominators, Step by Step



$\Gamma(1) = \{0, 1, 2\} \cap \{0, 3, 4, 1\} = \{0, 1\}$ $\quad$ 1 is already in W

Dominators, Step by Step



Join ∩: a **must** analysis. Facts that hold on *every* path.

Contents

Four Analyses

Two directions, two joins: four classic analyses.

	Join $\cup$ (may)	**Join** $\cap$ (must)
forward	reaching definitions	available expressions
backward	live variables	very busy expressions

Four Analyses

Two directions, two joins: four classic analyses.

	Join $\cup$ (may)	**Join** $\cap$ (must)
forward	reaching definitions	available expressions
backward	live variables	very busy expressions

- **Reaching definitions**: which assignment may have written a variable's value?
- **Live variables**: might a variable's current value still be read?
- **Available expressions**: may an expression's result be reused?
- **Very busy expressions**: will an expression be computed on every path ahead?

Reaching Definitions

Which assignment may have written the value of a variable here?

```
def f(n):
    i = 0               # i flows from here
    while i < n:        # i comes from two places
        i = i + 1       # and from here
    return i
```

Reaching Definitions

Which assignment may have written the value of a variable here?

```
def f(n):
    i = 0               # i flows from here
    while i < n:        # i comes from two places
        i = i + 1       # and from here
    return i
```

At **every point**, for **every variable**:

from which **assignment** did its value flow in?

Reaching Definitions

- **Data**: $\mathbb{D} = \mathcal{P}(\text{Var} \times (E \uplus \{\mathtt{p}\}))$
  ($\mathtt{x}_2$: $\mathtt{x}$ was defined on edge ℓ_2. $\mathtt{x}_\mathtt{p}$: $\mathtt{x}$ is a parameter.)

- **Join**: $\cup$ (**may**: true on some path)

- **Transfer**: for $\ell = p \xrightarrow{x\,\mathtt{:=}\,e} q$,

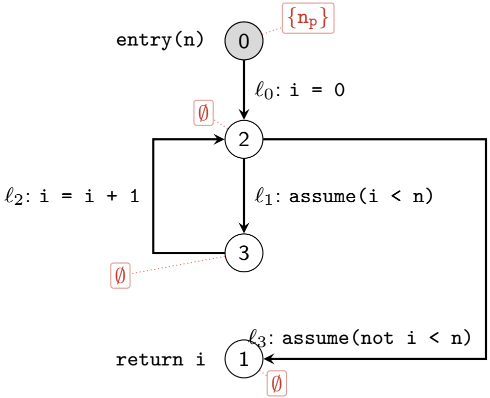
  $$\mathsf{kill}(\ell) = \{(x, d) \mid d \in E \uplus \{\mathtt{p}\}\} \qquad \mathsf{gen}(\ell) = \{(x, \ell)\}$$

  and $\mathsf{kill}(\ell) = \mathsf{gen}(\ell) = \emptyset$ for any other command.

- **Direction**: forward

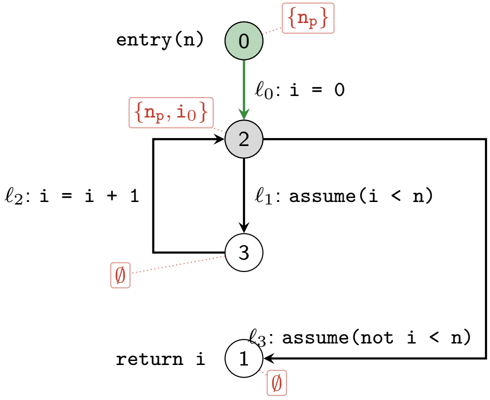
- **Initial**: $\Gamma_0(\mathtt{entry}) = \{(x, \mathtt{p}) \mid x \text{ a parameter}\}$, $\Gamma_0(p) = \emptyset$ otherwise

Reaching Definitions, Step by Step



Data $\mathbb{D} = \mathcal{P}(\mathsf{Var} \times (E \uplus \{\mathtt{p}\}))$ **Initial** $\Gamma_0(\mathtt{entry}) = \{\mathtt{n_p}\}$, $\Gamma_0(p) = \emptyset$ otherwise

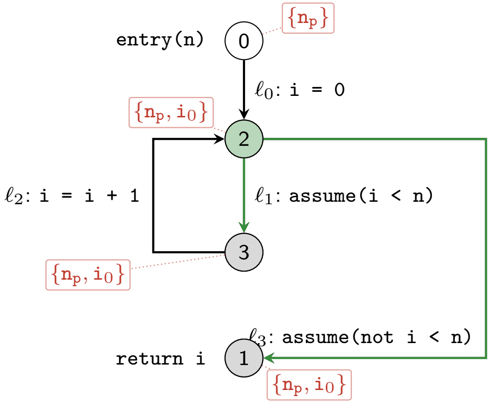
Reaching Definitions, Step by Step



$\mathsf{kill}(\ell) = \{(x, d) \mid d \in E \uplus \{\mathtt{p}\}\}$ $\mathsf{gen}(\ell) = \{(x, \ell)\}$ for $\ell = p \xrightarrow{x := e} q$ **Join** $\sqcup = \cup$

Direction forward $\mathsf{kill}(\ell_0) = \{\mathtt{i}_*\}$ $\mathsf{gen}(\ell_0) = \{\mathtt{i}_0\}$

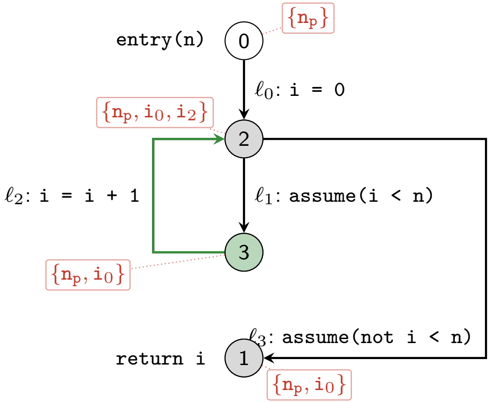
Reaching Definitions, Step by Step



$\mathsf{kill}(\ell) = \{(x, d) \mid d \in E \uplus \{\mathrm{p}\}\}$ $\quad \mathsf{gen}(\ell) = \{(x, \ell)\}$ $\quad$ for $\ell = p \xrightarrow{x:=e} q$ $\quad$ **Join** $\sqcup = \cup$

$\texttt{assume}$: $\mathsf{kill}(\ell_1) = \mathsf{gen}(\ell_1) = \emptyset \qquad \mathsf{kill}(\ell_3) = \mathsf{gen}(\ell_3) = \emptyset$

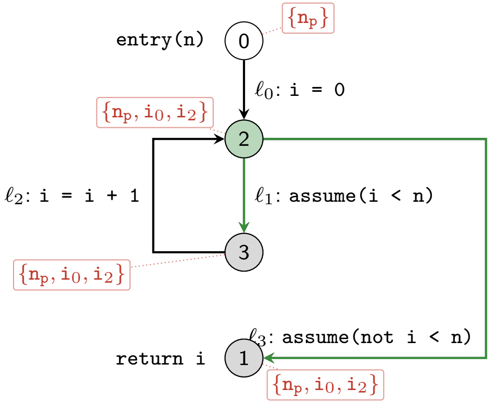
Reaching Definitions, Step by Step



$\text{kill}(\ell) = \{(x, d) \mid d \in E \uplus \{\mathtt{p}\}\}$ $\quad$ $\text{gen}(\ell) = \{(x, \ell)\}$ $\quad$ for $\ell = p \xrightarrow{x := e} q$ $\quad$ **Join** $\sqcup = \cup$

$$\text{kill}(\ell_2) = \{\mathtt{i}_*\} \qquad \text{gen}(\ell_2) = \{\mathtt{i}_2\}$$

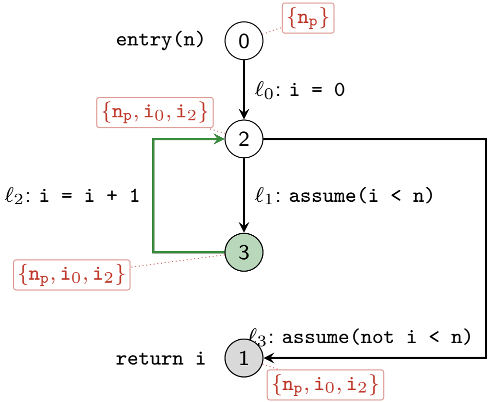
Reaching Definitions, Step by Step



$\mathsf{kill}(\ell) = \{(x, d) \mid d \in E \uplus \{\mathtt{p}\}\}$ $\quad\mathsf{gen}(\ell) = \{(x, \ell)\}$ $\quad$ for $\ell = p \xrightarrow{x:=e} q$ $\quad$ **Join** $\sqcup = \cup$

$\mathtt{assume}$: $\mathsf{kill}(\ell_1) = \mathsf{gen}(\ell_1) = \emptyset \qquad \mathsf{kill}(\ell_3) = \mathsf{gen}(\ell_3) = \emptyset$

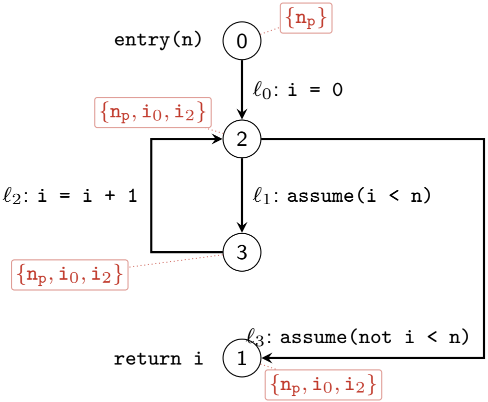
Reaching Definitions, Step by Step



$\mathsf{kill}(\ell) = \{(x, d) \mid d \in E \uplus \{\mathtt{p}\}\}$ $\quad \mathsf{gen}(\ell) = \{(x, \ell)\}$ $\quad$ for $\ell = p \xrightarrow{x := e} q$ $\quad$ **Join** $\sqcup = \cup$

$\mathsf{kill}(\ell_2) = \{\mathtt{i}_*\}$ $\quad \mathsf{gen}(\ell_2) = \{\mathtt{i}_2\}$ $\quad \Gamma(2)$ unchanged

Reaching Definitions, Step by Step



Join $\cup$: a **may** analysis. Facts that hold on *some* path.

At 2, `i` may come from ℓ_0 (first round) or ℓ_2 (later rounds).

Live Variables

Might the current value of a variable still be read later?

```
def f(a, b):
    x = a * 2          # dead: overwritten before any read
    y = b * 2          # live: one branch reads it
    if a < b:
        x = y
    else:
        x = b
    return x
```

Live Variables

Might the current value of a variable still be read later?

```
def f(a, b):
    x = a * 2           # dead: overwritten before any read
    y = b * 2           # live: one branch reads it
    if a < b:
        x = y
    else:
        x = b
    return x
```

At **every point**, for **every variable**:
may its **current value** still be **read**?

Live Variables

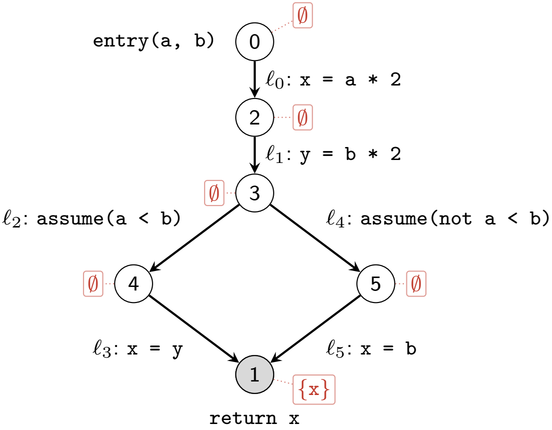
- **Data**: $\mathbb{D} = \mathcal{P}(\mathsf{Var})$
- **Join**: $\cup$ (**may**: true on some path)
- **Transfer**: for $\ell = p \xrightarrow{c} q$,

$$\mathsf{kill}(\ell) = \mathsf{defs}(c) \qquad \mathsf{gen}(\ell) = \mathsf{uses}(c)$$

  (defs(c): the variables c writes. uses(c): the variables c reads.)

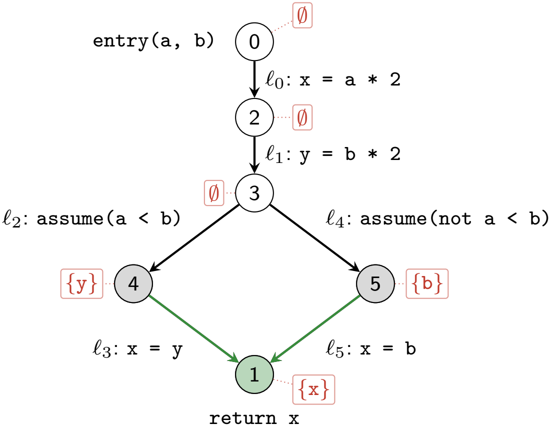
- **Direction**: backward
- **Initial**: $\Gamma_0(\texttt{exit}) = \mathsf{vars}(\text{the returned expression})$, $\Gamma_0(p) = \emptyset$ otherwise

Live Variables, Step by Step



Data $\mathbb{D} = \mathcal{P}(\mathrm{Var})$ **Initial** $\Gamma_0(\texttt{exit}) = \{\texttt{x}\}$ from `return x`, $\Gamma_0(p) = \emptyset$ otherwise

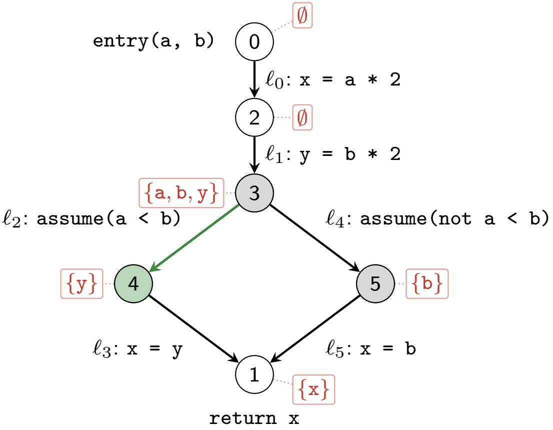
Live Variables, Step by Step



$\mathsf{kill}(\ell) = \mathsf{defs}(c)$ $\mathsf{gen}(\ell) = \mathsf{uses}(c)$ for $\ell = p \xrightarrow{c} q$ **Join** $\sqcup = \cup$

Direction backward $\mathsf{kill}(\ell_3) = \mathsf{kill}(\ell_5) = \{\mathtt{x}\}$ $\mathsf{gen}(\ell_3) = \{\mathtt{y}\}$, $\mathsf{gen}(\ell_5) = \{\mathtt{b}\}$

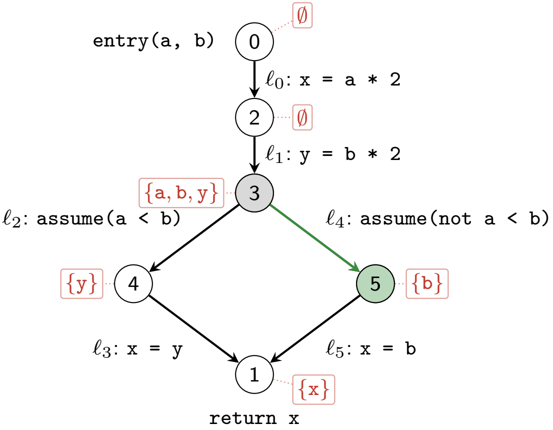
Live Variables, Step by Step



$\mathsf{kill}(\ell) = \mathsf{defs}(c)$ $\mathsf{gen}(\ell) = \mathsf{uses}(c)$ for $\ell = p \xrightarrow{c} q$ **Join** $\sqcup = \cup$

$$\mathsf{kill}(\ell_2) = \emptyset \qquad \mathsf{gen}(\ell_2) = \{\mathsf{a}, \mathsf{b}\}$$

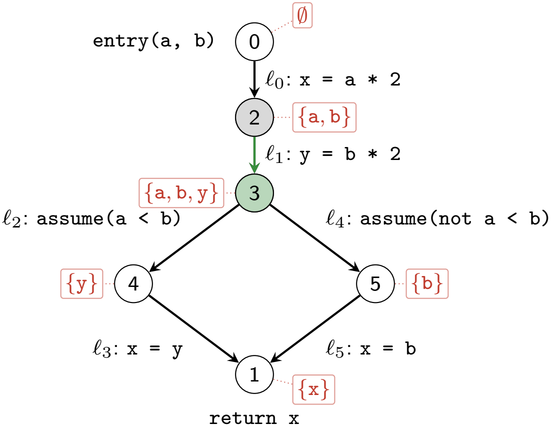
Live Variables, Step by Step



kill(ℓ) = defs(c)   gen(ℓ) = uses(c)   for $\ell = p \xrightarrow{c} q$ **Join** $\sqcup = \cup$

kill(ℓ_4) $= \emptyset$ gen(ℓ_4) $= \{\mathsf{a}, \mathsf{b}\}$ $\Gamma(3)$ unchanged

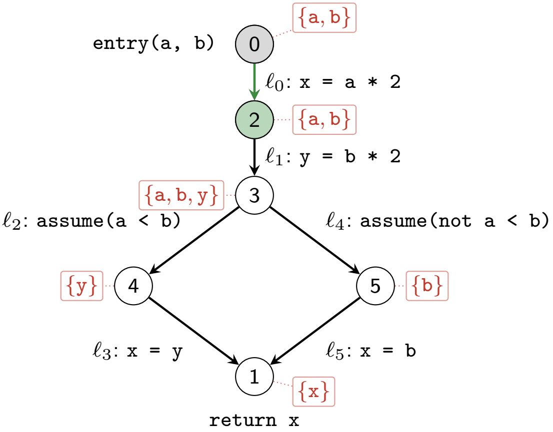
Live Variables, Step by Step



kill(ℓ) = defs(c) gen(ℓ) = uses(c) for $\ell = p \xrightarrow{c} q$ **Join** $\sqcup = \cup$

kill$(\ell_1) = \{$y$\}$ gen$(\ell_1) = \{$b$\}$

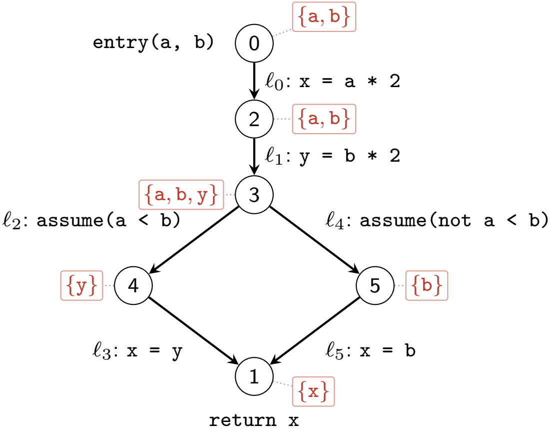
Live Variables, Step by Step



$\text{kill}(\ell) = \text{defs}(c) \quad \text{gen}(\ell) = \text{uses}(c) \quad \text{for } \ell = p \xrightarrow{c} q \quad$ **Join** $\sqcup = \cup$

$$\text{kill}(\ell_0) = \{\texttt{x}\} \qquad \text{gen}(\ell_0) = \{\texttt{a}\}$$

Live Variables, Step by Step



Join ∪: a **may** analysis. Facts that hold on *some* path.

`x` is dead at 2: `x = a * 2` can go. `y` is live at 3: one branch reads it.

Available Expressions

May the result of an expression be reused here?

```
def f(a, b, c):
    x = (a + b) + (c * 2)  # computes a + b and c * 2
    if a < b:
        a = 0              # a changes on this path
    y = (a + b) + 3        # a + b: must recompute
    z = 1 + (c * 2)        # c * 2: can reuse
    return y + z
```

Available Expressions

May the result of an expression be reused here?

```
def f(a, b, c):
    x = (a + b) + (c * 2)  # computes a + b and c * 2
    if a < b:
        a = 0              # a changes on this path
    y = (a + b) + 3        # a + b: must recompute
    z = 1 + (c * 2)        # c * 2: can reuse
    return y + z
```

At **every point**: which **expressions** were computed on **every path**, with no operand changed since?

Available Expressions

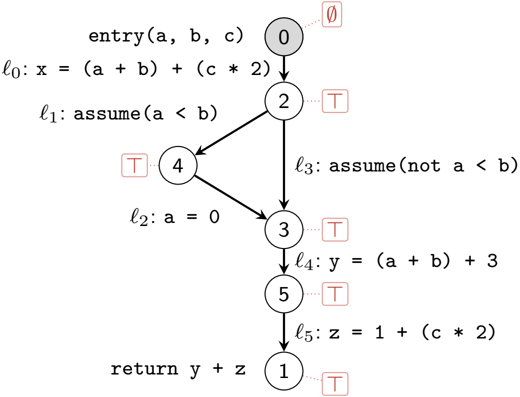
PLRG

- **Data**: $\mathbb{D} = \mathcal{P}(\mathsf{Expr})$
- **Join**: $\cap$ (**must**: true on every path)
- **Transfer**: for $\ell = p \xrightarrow{c} q$,

  $$\mathsf{kill}(\ell) = \{e \mid \mathsf{vars}(e) \cap \mathsf{defs}(c) \neq \emptyset\} \qquad \mathsf{gen}(\ell) = \mathsf{exprs}(c) \setminus \mathsf{kill}(\ell)$$

  ($\mathsf{exprs}(c)$: the expressions c evaluates.)
- **Direction**: forward
- **Initial**: $\Gamma_0(\mathtt{entry}) = \emptyset$, $\Gamma_0(p) = \mathsf{Expr}$ otherwise

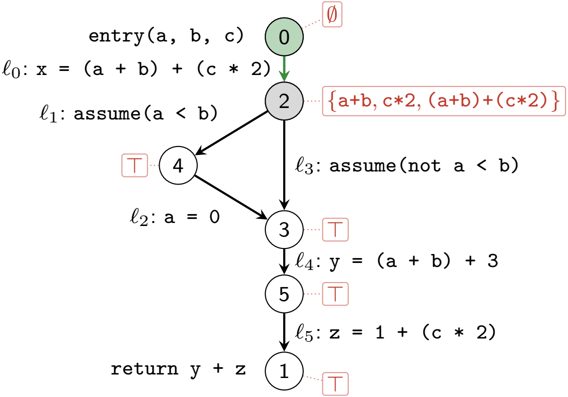
Available Expressions, Step by Step



Data $\mathbb{D} = \mathcal{P}(\text{Expr})$ **Initial** $\Gamma_0(\texttt{entry}) = \emptyset$, $\Gamma_0(p) = \top$ otherwise

($\top$ = Expr: every expression in the program)

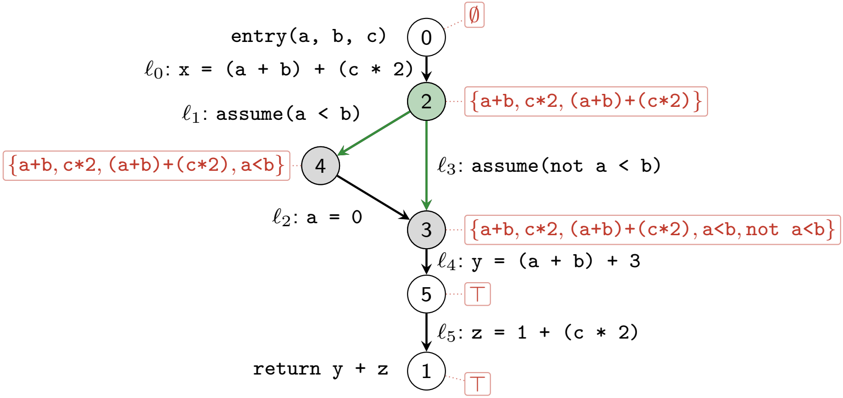
Available Expressions, Step by Step



$\text{kill}(\ell) = \{e \mid \text{vars}(e) \cap \text{defs}(c) \neq \emptyset\}$ $\text{gen}(\ell) = \text{exprs}(c) \setminus \text{kill}(\ell)$ for $\ell = p \xrightarrow{c} q$ **Join** $\sqcup = \cap$

Direction forward $\text{kill}(\ell_0) = \emptyset$ $\text{gen}(\ell_0) = \{\texttt{a+b}, \texttt{c*2}, \texttt{(a+b)+(c*2)}\}$

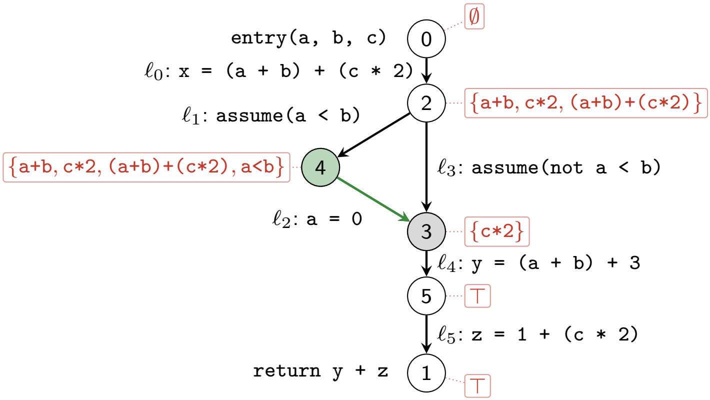
Available Expressions, Step by Step



$\mathsf{kill}(\ell) = \{e \mid \mathsf{vars}(e) \cap \mathsf{defs}(c) \neq \emptyset\}$ $\quad \mathsf{gen}(\ell) = \mathsf{exprs}(c) \setminus \mathsf{kill}(\ell)$ $\quad$ for $\ell = p \xrightarrow{c} q$ $\quad$ **Join** $\sqcup = \cap$

$$\mathsf{kill}(\ell_1) = \mathsf{kill}(\ell_3) = \emptyset \qquad \mathsf{gen}(\ell_1) = \{\texttt{a<b}\},\ \mathsf{gen}(\ell_3) = \{\texttt{a<b}, \texttt{not a<b}\}$$

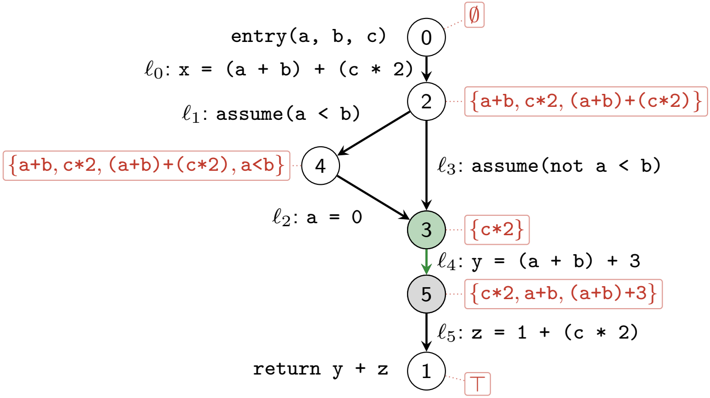
Available Expressions, Step by Step



$\text{kill}(\ell) = \{e \mid \text{vars}(e) \cap \text{defs}(c) \neq \emptyset\}$ $\quad \text{gen}(\ell) = \text{exprs}(c) \setminus \text{kill}(\ell)$ $\quad$ for $\ell = p \xrightarrow{c} q$ $\quad$ **Join** $\sqcup = \cap$

$$\text{kill}(\ell_2) = \{\texttt{a+b}, \texttt{(a+b)+3}, \texttt{(a+b)+(c*2)}, \texttt{a<b}, \texttt{not a<b}\} \qquad \text{gen}(\ell_2) = \emptyset$$

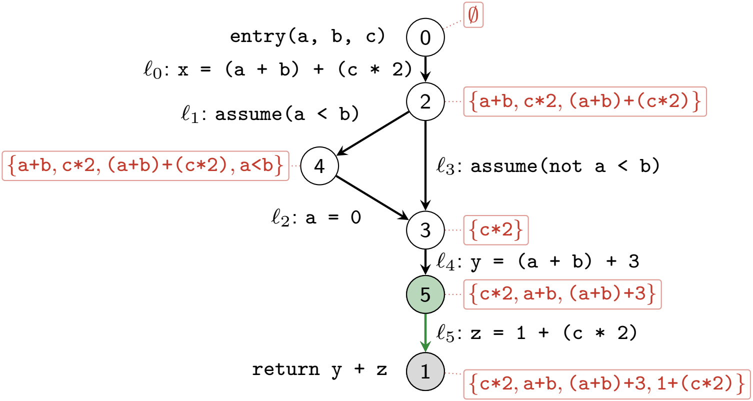
Available Expressions, Step by Step



$\text{kill}(\ell) = \{e \mid \text{vars}(e) \cap \text{defs}(c) \neq \emptyset\}$ $\quad$ $\text{gen}(\ell) = \text{exprs}(c) \setminus \text{kill}(\ell)$ $\quad$ for $\ell = p \xrightarrow{c} q$ $\quad$ **Join** $\sqcup = \cap$

$$\text{kill}(\ell_4) = \{\texttt{y+z}\} \qquad \text{gen}(\ell_4) = \{\texttt{a+b, (a+b)+3}\}$$

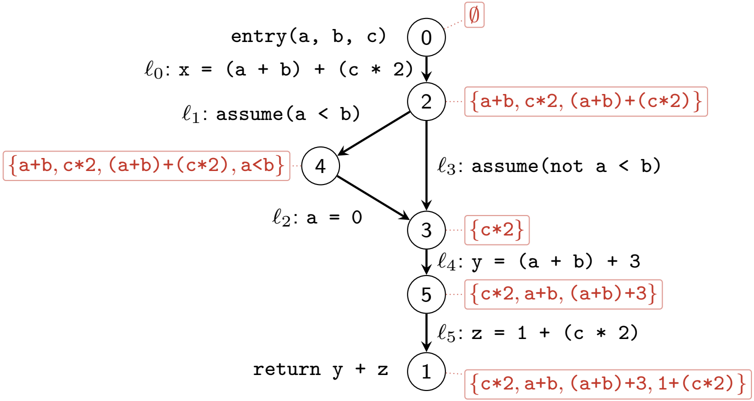
Available Expressions, Step by Step



$\mathsf{kill}(\ell) = \{e \mid \mathsf{vars}(e) \cap \mathsf{defs}(c) \neq \emptyset\}$ $\quad \mathsf{gen}(\ell) = \mathsf{exprs}(c) \setminus \mathsf{kill}(\ell)$ $\quad$ for $\ell = p \xrightarrow{c} q$ $\quad$ **Join** $\sqcup = \cap$

$$\mathsf{kill}(\ell_5) = \{\texttt{y+z}\} \qquad \mathsf{gen}(\ell_5) = \{\texttt{c*2}, \texttt{1+(c*2)}\}$$

Available Expressions, Step by Step



Join $\cap$: a **must** analysis. Facts that hold on *every* path.

At 3, c*2 is still available and a+b is not: reuse one, recompute the other.

Very Busy Expressions

Will an expression be computed on every path from here?

```
def f(a, b, c):
    if c < 0:
        x = (a * b) + (b + 1)
    else:
        a = 1                       # a changes: kills a * b
        x = (a * b) + (b + 1)
    return x
```

Very Busy Expressions

Will an expression be computed on every path from here?

```
def f(a, b, c):
    if c < 0:
        x = (a * b) + (b + 1)
    else:
        a = 1                       # a changes: kills a * b
        x = (a * b) + (b + 1)
    return x
```

At **every point**: which **expressions** will **every path** compute next, before an operand changes?

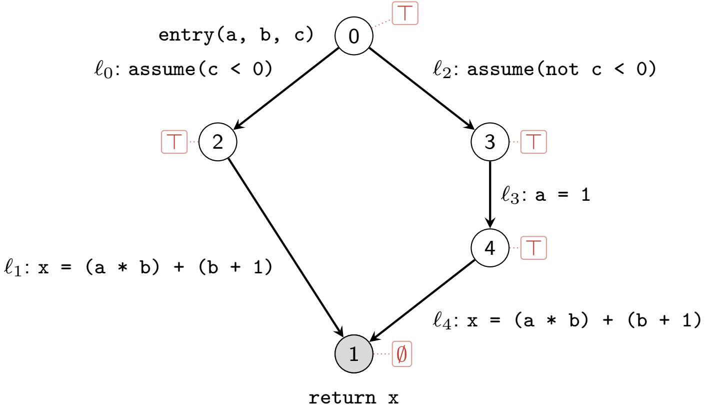
Very Busy Expressions

- **Data**: $\mathbb{D} = \mathcal{P}(\mathsf{Expr})$
- **Join**: $\cap$ (**must**: true on every path)
- **Transfer**: for $\ell = p \xrightarrow{c} q$,

$$\mathsf{kill}(\ell) = \{e \mid \mathsf{vars}(e) \cap \mathsf{defs}(c) \neq \emptyset\} \qquad \mathsf{gen}(\ell) = \mathsf{exprs}(c)$$

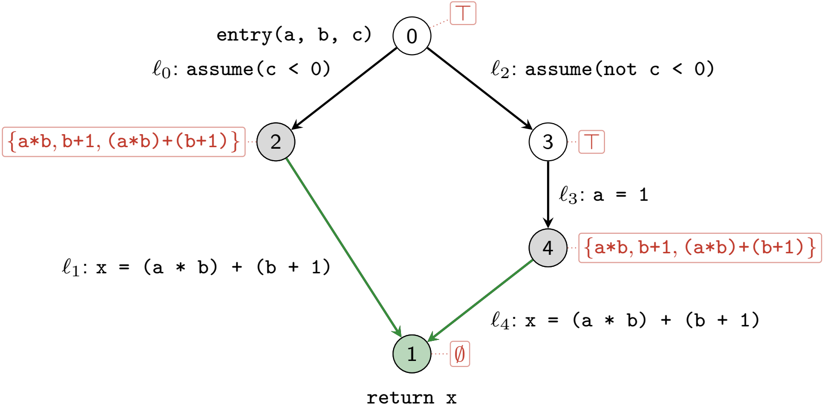
- **Direction**: backward
- **Initial**: $\Gamma_0(\texttt{exit}) = \mathsf{exprs}(\text{the returned expression})$, $\Gamma_0(p) = \mathsf{Expr}$ otherwise

Very Busy Expressions, Step by Step



Data $\mathbb{D} = \mathcal{P}(\mathsf{Expr})$ **Initial** $\Gamma_0(\texttt{exit}) = \emptyset$ from `return x`, $\Gamma_0(p) = \top$ otherwise ($\top = \mathsf{Expr}$: every expression in the program)

Very Busy Expressions, Step by Step

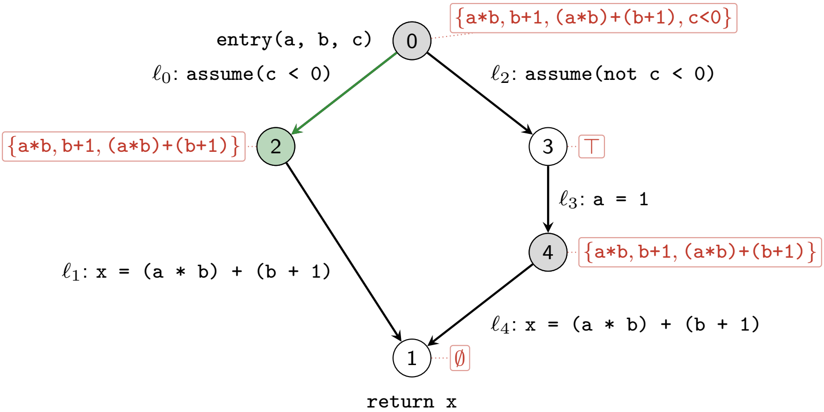


$\mathsf{kill}(\ell) = \{e \mid \mathsf{vars}(e) \cap \mathsf{defs}(c) \neq \emptyset\}$ $\mathsf{gen}(\ell) = \mathsf{exprs}(c)$ for $\ell = p \xrightarrow{c} q$ **Join** $\sqcup = \cap$

Direction backward $\mathsf{kill}(\ell_1) = \mathsf{kill}(\ell_4) = \emptyset$

$\mathsf{gen}(\ell_1) = \mathsf{gen}(\ell_4) = \{\texttt{a*b}, \texttt{b+1}, \texttt{(a*b)+(b+1)}\}$

Very Busy Expressions, Step by Step

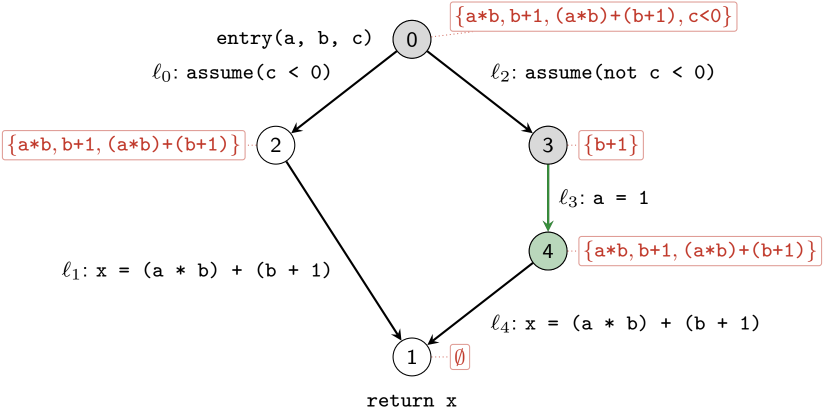


$\mathsf{kill}(\ell) = \{e \mid \mathsf{vars}(e) \cap \mathsf{defs}(c) \neq \emptyset\}$ $\quad \mathsf{gen}(\ell) = \mathsf{exprs}(c)$ $\quad$ for $\ell = p \xrightarrow{c} q$ $\quad$ **Join** $\sqcup = \cap$

$$\mathsf{kill}(\ell_0) = \emptyset \qquad \mathsf{gen}(\ell_0) = \{\texttt{c<0}\}$$

Very Busy Expressions, Step by Step

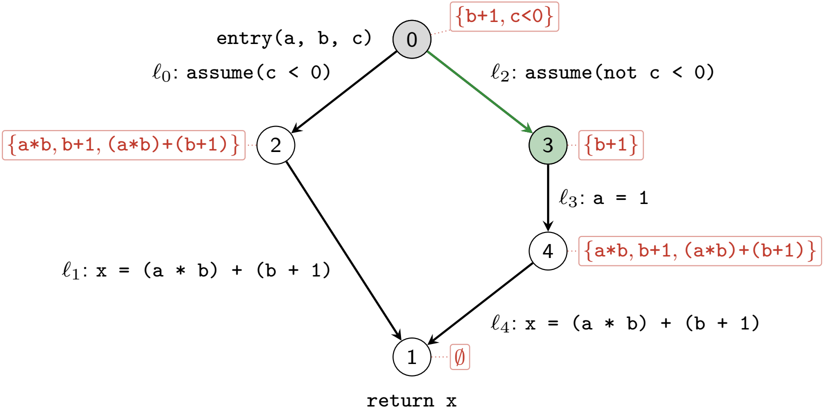
PLRG



$\mathsf{kill}(\ell) = \{e \mid \mathsf{vars}(e) \cap \mathsf{defs}(c) \neq \emptyset\}$ $\quad \mathsf{gen}(\ell) = \mathsf{exprs}(c)$ $\quad$ for $\ell = p \xrightarrow{c} q$ $\quad$ **Join** $\sqcup = \cap$

$$\mathsf{kill}(\ell_3) = \{\texttt{a*b}, \texttt{(a*b)+(b+1)}\} \qquad \mathsf{gen}(\ell_3) = \emptyset$$

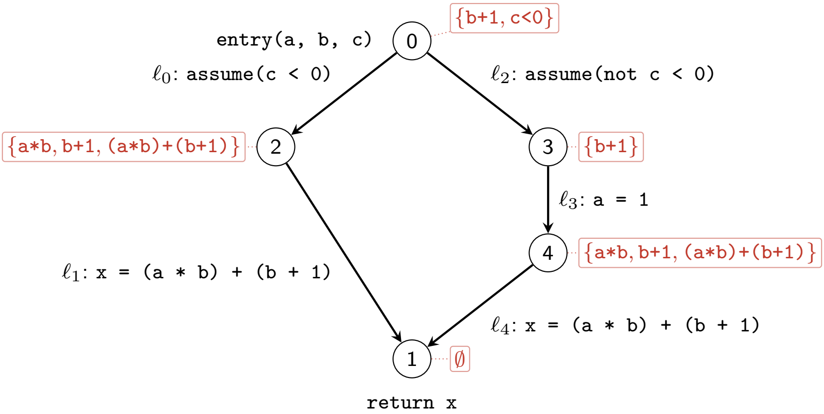
Very Busy Expressions, Step by Step



$\mathsf{kill}(\ell) = \{e \mid \mathsf{vars}(e) \cap \mathsf{defs}(c) \neq \emptyset\}$ $\mathsf{gen}(\ell) = \mathsf{exprs}(c)$ for $\ell = p \xrightarrow{c} q$ **Join** $\sqcup = \cap$

$$\mathsf{kill}(\ell_2) = \emptyset \qquad \mathsf{gen}(\ell_2) = \{\texttt{c<0}, \texttt{not c<0}\}$$

Very Busy Expressions, Step by Step



Join $\cap$: a **must** analysis. Facts that hold on *every* path.

`b+1` is very busy at `entry`: hoist it. `a*b` is not: one branch changes `a` first.

Contents

In Python – An Analysis

```
Data = frozenset

PARAM = -1          # the "edge" that defines the parameters: the call itself

@dataclass(frozen=True)
class Analysis:
    forward: bool
    bottom: Data                                        # initial guess everywhere
    init: Data                                          # the data at the start point
    join: Callable[[Data, Data], Data]
    transfer: Callable[[Edge, Data], Data]              # what one edge does
```

- `bottom` and `join`: the data and $\sqcup$. `init`: Γ_0 at the start point.
- `transfer`: f_ℓ, given the Edge. `forward`: which F.

In Python – The Engine

```
def solve(g: CFG, a: Analysis) -> dict[NodeId, Data]:
    """Least fixpoint of the dataflow equations: one set per program point.

        forward:   data(q) = join of transfer(e, data(p)) for every edge p -> q
        backward:  data(p) = join of transfer(e, data(q)) for every edge p -> q
    """
    start = g.entry if a.forward else g.exit
    data = {p: a.bottom for p in g}
    data[start] = a.init
    work = Worklist([start])
    while work:
        p = work.pop()
        # for every edge out of p in the direction of a, to a point q:
        #   data[q] = a.join(data[q], a.transfer(e, data[p]))
        #   and push q if that changed it, or if q is newly reached
    return data
```

- The dominators loop, reading its four parameters from a.

In Python – One Configuration

```
def dominators(g: CFG) -> Analysis:
    """Which points must run before this one?"""
    def transfer(e: Edge, s: Data) -> Data:
        return s | {e.dst}                          # kill nothing, gen q

    return Analysis(True, frozenset(g.nodes), frozenset({g.entry}),
                    frozenset.intersection, transfer)
```

- The dominators table, as data: forward, bottom is N, init is $\{\texttt{entry}\}$, join is $\cap$.
- One line of content: $f_\ell(X) = X \cup \{q\}$.
- The other three analyses: change these five values.

Homework 1

- Fill in `dataflow.py`: the engine, **reaching definitions**, **live variables**, **available expressions**.
- Everything you need, and the tests that grade you:

  github.com/ku-plrg-classroom/py-dataflow
- Due October 12 on the **LMS**. Submit `dataflow.py` only.

Summary

Next Lecture

- Taint Analysis

Jihyeok Park
jihyeok_park@korea.ac.kr
https://plrg.korea.ac.kr