Direction of dataflow analysis

In what order are constraints solved, in general?

Constraints are declarative, not directional/procedural, so may require mixing forward & backward solving, or other more global solution methods

But often constraints can be solved by (directional) propagation & iteration

- may be **forward** or **backward** propagation of info
- topological traversals of acyclic subgraphs minimize analysis time

Directional constraints often called **flow functions**

- often written as functions on input info to compute output

$RD_{s:\,\texttt{x := ...}}(in) = in - \{x \to s' \mid \forall s'\} \cup \{x \to s\}$

$RD_{s:\,\texttt{*p := ...}}(in) = in \cup \{x \to s \mid \forall x \in \text{may-point-to}(p)\}$

GEN and KILL sets

For even more structure,
can often think of flow functions in terms of each's GEN set and KILL set

- GEN = new information added
- KILL = old information removed

Then

$F_{instr}(in) = in - KILL_{instr} \cup GEN_{instr}$

E.g., for reaching defs:

$RD_{s:\,\texttt{x := ...}}(in) = in - \{x \to s' \mid \forall s'\} \cup \{x \to s\}$

$RD_{s:\,\texttt{*p := ...}}(in) = in \qquad \cup \{x \to s \mid \forall x \in \text{mpt}(p)\}$

Bit vectors

For maximum efficiency,
can sometimes represent info/KILL/GEN sets as **bit vectors**

- if can express abstractly as set of things (e.g. statements, vars), drawn from a statically known set of things, each thing getting a statically determined bit position
- bitvector encodes **characteristic function** of set

E.g., for reaching defs:

info = bitvector over statements,
each stmt getting a distinct bit position

- statement implies which variable is defined

Bit vectors compactly represent sets

Bit-vector operations efficiently perform set difference & union

Flow function may be able to be represented simply by a pair of bit vectors, if they don't depend on input bit vector

- can merge the KILL and GEN bit vectors of a whole basic block of instructions into a single overall KILL and GEN set, for faster iterating

Another example: constant propagation

What info computed for each program point?

I is a conservative approximation to true info I_{true} iff:

Direction of analysis?

Initial info?

$CP_{\texttt{x := } N}$:

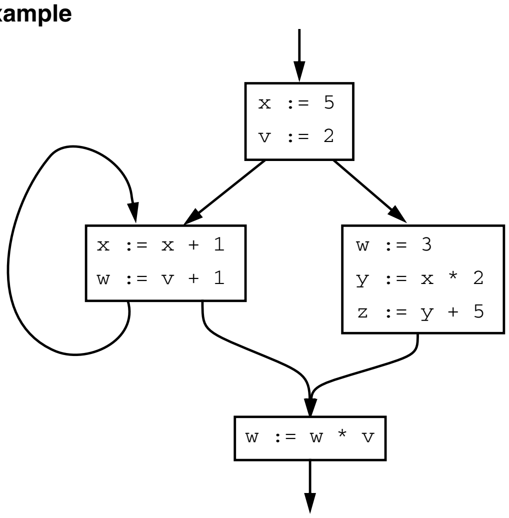
$CP_{\texttt{x := y + z}}$:

$CP_{\texttt{*p := *q + *r}}$:

Merge function?

Can use bit vectors?

Example



May vs. must info

Some kinds of info imply guarantees: **must** info
Some kinds of info imply possibilities: **may** info
- the complement of **may** info is **must not** info

	May	**Must**
desired info	small set	big set
safe	overly big set	overly small set
GEN	add everything that might be true	add only if guaranteed true
KILL	remove only if guaranteed wrong	remove everything possibly wrong
MERGE	$\cup$	$\cap$

Another example: live variables

Want the set of variables that are **live** at each pt. in program
- live: *might* be used *later* in the program

Supports dead assignment elimination, register allocation

What info computed for each program point?
May or must info?
I is a conservative approximation to true info I_{true} iff:

Direction of analysis?
Initial info, at what program point(s)?

$LV_{\texttt{x := y + z}}$:

$LV_{\texttt{*p := *q + *r}}$:

Merge function?

Can use bit vectors?

Example