Liveness Analysis
propagating liveness

• Suppose we have a set of variables that are live at a particular point in the program.
• What does it mean to “execute” a statement backwards?

\[
\{???\}
\]

\[
A = B \times C
\]

\[
\{A, D\}
\]
Suppose we have a set of variables that are live at a particular point in the program.

What does it mean to “execute” a statement backwards?

\[
\{B, C, D\}
\]

\[
A = B \times C
\]

\[
\{A, D\}
\]
Suppose we have a set of variables that are live at a particular point in the program.

What does it mean to “execute” a statement backwards?

propagating liveness

```
\{B, C, D\} \quad L_{in}

A = B \times C

L_{in} = (L_{out} - K) \cup G

\{A, D\} \quad L_{out}
```
Suppose we have a set of variables that are live at a particular point in the program.

What does it mean to “execute” a statement backwards?

\[ A = B \times C \]

\[ \mathcal{L}_{\text{in}} = \mathcal{L}_{\text{out}} \cup \mathcal{K} \]

variables “generated”, or used, by a statement

variables “killed”, or defined, by a statement
What is live in this code?

1: \( A = B + C \)  \{B, C\}
2: \( C = A + B \)  \{A, B\}
3: \( T1 = B + C \)  \{A, B, C\}
4: \( T2 = T1 + C \)  \{A, B, C, T1\}
5: \( D = T2 \)  \{A, B, C, D\}
6: \( E = A + B \)  \{C, D, E\}
7: \( B = E + D \)  \{B, C, D\}
8: \( A = C + D \)  \{B, C, D\}
9: \( T3 = A + B \)  \{A, B\}
10: WRITE(T3)  \{T3\}
what about aliasing?

• Aliasing, as usual is a problem

• Reminder: compilers must be conservative

• Liveness is a *may* property → OK to say something is live when it isn’t
  • This *may* be used in the future (even if it really won’t be)

• Deal with aliasing by being conservative:
  • A variable stops being live when it is written to
  • Only *kill* variables that are *definitely* written to
next: finding dead code