Phases of a compiler

![Diagram of compiler phases]

**Optimizations**

*Figure 1.6, page 5 of text*
9.2.3 Data-flow schemas on basic blocks

"...data-flow equations usually do not have a unique solution. Our goal is to find the most 'precise' solution that satisfies the two sets of constraints: control-flow and transfer constraints. That is, we need a solution that encourages valid code improvements, but does not justify unsafe transformations..."

[p. 601]
9.2.4 Reaching definitions

"A definition d reaches a point p if there is a path from the point immediately following d to p, such that d is not 'killed' along that path." [p. 601]

"We kill a definition of a variable x if there is any other definition of x anywhere along the path." [p. 601]
9.2.4 Reaching definitions

“A definition of a variable \( x \) is a statement that assigns, or may assign, a value to \( x \).”

What is meant by “may assign”? 
9.2.4 Reaching definitions

"Procedure parameters, array accesses, and indirect references all may have aliases, and it is not easy to tell if a statement is referring to a particular variable x." [p. 601]

"Program analysis must be conservative" [p. 601]
Transfer equations for reaching definitions

For this definition:

\[ d: u = v + w \]

The transfer equation is:

\[ f_d(\sigma) = \text{gend} \cup (\sigma - \text{kill}_d) \]

where \( \text{gend} = \{d\} \). \( \text{kill}_d \) is the set of all other definitions of \( u \) in the program.

\( \sigma \) is a data-flow value

The argument of a transfer function is a data-flow value, which "represents an abstraction of the set of all possible program states that can be observed for that point." [p. 599]

Recall too that a program state consists of all the variables in the program along with their current values.
Figure 9.13 (p. 604)

\[
\begin{align*}
d1: i &= m - 1 \\
d2: j &= n \\
d3: a &= u1 \\
d4: i &= i + 1 \\
d5: j &= j - 1 \\
d6: a &= u2 \\
d7: i &= u3 \\
g_{B1} &= \{ ? \} \\
k_{B1} &= \{ ? \} \\
g_{B2} &= \{ ? \} \\
k_{B2} &= \{ ? \} \\
g_{B3} &= \{ ? \} \\
k_{B3} &= \{ ? \} \\
g_{B4} &= \{ ? \} \\
k_{B4} &= \{ ? \}
\end{align*}
\]
Figure 9.13 (p. 604)

\[ d_1: i = m - 1 \]
\[ d_2: j = n \]
\[ d_3: a = u_1 \]

\[ d_4: i = i + 1 \]
\[ d_5: j = j - 1 \]

\[ d_6: a = u_2 \]

\[ d_7: i = u_3 \]
Figure 9.13 (p. 604)

\begin{itemize}
  \item d1: \( i = m - 1 \)
  \item d2: \( j = n \)
  \item d3: \( a = u_1 \)
  \item d4: \( i = i + 1 \)
  \item d5: \( j = j - 1 \)
  \item d6: \( a = u_2 \)
  \item d7: \( i = u_3 \)
\end{itemize}

\begin{itemize}
  \item gen\(_{B1}\) = \{d1, d2, d3\}
  \item kill\(_{B1}\) = \{d4, d5, d6, d7\}
  \item gen\(_{B2}\) = \{?\}
  \item kill\(_{B2}\) = \{?\}
  \item gen\(_{B3}\) = \{?\}
  \item kill\(_{B3}\) = \{?\}
  \item gen\(_{B4}\) = \{?\}
  \item kill\(_{B4}\) = \{?\}
\end{itemize}
Figure 9.13 (p. 604)

Q: Why kill d4 - d7 here, since they are not on a path to B1?

**B1**

- gen\_B1 = \{ d1, d2, d3 \}
- kill\_B1 = \{ d4, d5, d6, d7 \}

**B2**

- gen\_B2 = \{ ? \}
- kill\_B2 = \{ ? \}

**B3**

- gen\_B3 = \{ ? \}
- kill\_B3 = \{ ? \}

**B4**

- gen\_B4 = \{ ? \}
- kill\_B4 = \{ ? \}
Figure 9.13 (p. 604)

\[ d1: i = m - 1 \]
\[ d2: j = n \]
\[ d3: a = u1 \]

\[ d4: i = i + 1 \]
\[ d5: j = j - 1 \]
\[ d6: a = u2 \]
\[ d7: i = u3 \]

\[ \text{gen}_B1 = \{ d1, d2, d3 \} \]
\[ \text{kill}_B1 = \{ d4, d5, d6, d7 \} \]

Q: Why kill d4 - d7 here, since they are not on a path to B1?

A: Here we are looking just at this block, and not trying to account for flow between blocks.

Inter-block flow is taken into account later.
Figure 9.13 (p. 604)

\[
\begin{align*}
  d1: i &= m - 1 \\
  d2: j &= n \\
  d3: a &= u1 \\
  d4: i &= i + 1 \\
  d5: j &= j - 1 \\
  d6: a &= u2 \\
  d7: i &= u3
\end{align*}
\]

\[
\begin{align*}
  \text{gen}_{B1} &= \{d1, d2, d3\} \\
  \text{kill}_{B1} &= \{d4, d5, d6, d7\} \\
  \text{gen}_{B2} &= \{?\} \\
  \text{kill}_{B2} &= \{?\} \\
  \text{gen}_{B3} &= \{?\} \\
  \text{kill}_{B3} &= \{?\} \\
  \text{gen}_{B4} &= \{?\} \\
  \text{kill}_{B4} &= \{?\}
\end{align*}
\]
Figure 9.13 (p. 604)

\[
\begin{align*}
\text{ENTRY} & \\
\text{B1} & \\
\text{d1: } i &= m - 1 \\
\text{d2: } j &= n \\
\text{d3: } a &= u1 \\
\text{gen}_{B1} &= \{ \text{d1, d2, d3} \} \\
\text{kill}_{B1} &= \{ \text{d4, d5, d6, d7} \} \\
\text{B2} & \\
\text{d4: } i &= i + 1 \\
\text{d5: } j &= j - 1 \\
\text{gen}_{B2} &= \{ \text{d4, d5} \} \\
\text{kill}_{B2} &= \{ \text{?} \} \\
\text{B3} & \\
\text{d6: } a &= u2 \\
\text{gen}_{B3} &= \{ \text{?} \} \\
\text{kill}_{B3} &= \{ \text{?} \} \\
\text{B4} & \\
\text{d7: } i &= u3 \\
\text{gen}_{B4} &= \{ \text{?} \} \\
\text{kill}_{B4} &= \{ \text{?} \} \\
\text{EXIT} &
\end{align*}
\]
Figure 9.13 (p. 604)

\begin{align*}
\text{ENTRY} & \\
& \quad \Downarrow \\
& B1 \\
& \quad \Downarrow \\
& B2 \\
& \quad \Downarrow \\
& B3 \\
& \quad \Downarrow \\
& B4 \\
\end{align*}

\begin{align*}
d1: & \quad i = m - 1 \\
d2: & \quad j = n \\
d3: & \quad a = u1 \\
d4: & \quad i = i + 1 \\
d5: & \quad j = j - 1 \\
d6: & \quad a = u2 \\
d7: & \quad i = u3 \\
\end{align*}

\begin{align*}
gen_{B1} & = \{ d1, d2, d3 \} \\
kill_{B1} & = \{ d4, d5, d6, d7 \} \\
gen_{B2} & = \{ d4, d5 \} \\
kill_{B2} & = \{ d1, d2, d7 \} \\
gen_{B3} & = \{ \, ? \, \} \\
kill_{B3} & = \{ \, ? \, \} \\
gen_{B4} & = \{ \, ? \, \} \\
kill_{B4} & = \{ \, ? \, \} \\
\end{align*}
Figure 9.13 (p. 604)

\begin{align*}
\text{d1: } & i = m - 1 \\
\text{d2: } & j = n \\
\text{d3: } & a = u_1 \\
\text{d4: } & i = i + 1 \\
\text{d5: } & j = j - 1 \\
\text{d6: } & a = u_2 \\
\text{d7: } & i = u_3
\end{align*}

\begin{align*}
\text{gen}_{B1} &= \{ \text{d1, d2, d3} \} \\
\text{kill}_{B1} &= \{ \text{d4, d5, d6, d7} \} \\
\text{gen}_{B2} &= \{ \text{d4, d5} \} \\
\text{kill}_{B2} &= \{ \text{d1, d2, d7} \} \\
\text{gen}_{B3} &= \{ \text{d6} \} \\
\text{kill}_{B3} &= \{ \text{?} \} \\
\text{gen}_{B4} &= \{ \text{?} \} \\
\text{kill}_{B4} &= \{ \text{?} \}
\end{align*}
Figure 9.13 (p. 604)

\[
\begin{align*}
\text{d1: } i &= m - 1 \\
\text{d2: } j &= n \\
\text{d3: } a &= u_1 \\
\text{d4: } i &= i + 1 \\
\text{d5: } j &= j - 1 \\
\text{d6: } a &= u_2 \\
\text{d7: } i &= u_3
\end{align*}
\]

\[\text{gen}_{B1} = \{ \text{d1, d2, d3} \} \]
\[\text{kill}_{B1} = \{ \text{d4, d5, d6, d7} \} \]

\[\text{gen}_{B2} = \{ \text{d4, d5} \} \]
\[\text{kill}_{B2} = \{ \text{d1, d2, d7} \} \]

\[\text{gen}_{B3} = \{ \text{d6} \} \]
\[\text{kill}_{B3} = \{ \text{d3} \} \]

\[\text{gen}_{B4} = \{ \text{?} \} \]
\[\text{kill}_{B4} = \{ \text{?} \} \]
Figure 9.13 (p. 604)

\[
\begin{align*}
\text{d1: } i &= m - 1 \\
\text{d2: } j &= n \\
\text{d3: } a &= u1
\end{align*}
\]

\[
\begin{align*}
\text{d4: } i &= i + 1 \\
\text{d5: } j &= j - 1
\end{align*}
\]

\[
\text{d6: } a &= u2
\]

\[
\text{d7: } i &= u3
\]

**gen**

\[B1 = \{ \text{d1, d2, d3} \} \]

\[\text{kill}_{B1} = \{ \text{d4, d5, d6, d7} \} \]

\[
\begin{align*}
\text{gen}_{B2} &= \{ \text{d4, d5} \} \\
\text{kill}_{B2} &= \{ \text{d1, d2, d7} \}
\end{align*}
\]

\[
\begin{align*}
\text{gen}_{B3} &= \{ \text{d6} \} \\
\text{kill}_{B3} &= \{ \text{d3} \}
\end{align*}
\]

\[
\begin{align*}
\text{gen}_{B4} &= \{ \text{d7} \} \\
\text{kill}_{B4} &= \{ ? \}
\end{align*}
\]

ENTRY

EXIT
Figure 9.13 (p. 604)

\[
\begin{align*}
\text{d1: } & i = m - 1 \\
\text{d2: } & j = n \\
\text{d3: } & a = u1 \\
\text{d4: } & i = i + 1 \\
\text{d5: } & j = j - 1 \\
\text{d6: } & a = u2 \\
\text{d7: } & i = u3
\end{align*}
\]

\[
\begin{align*}
\text{gen}_B1 &= \{ \text{d1, d2, d3} \} \\
\text{kill}_B1 &= \{ \text{d4, d5, d6, d7} \} \\
\text{gen}_B2 &= \{ \text{d4, d5} \} \\
\text{kill}_B2 &= \{ \text{d1, d2, d7} \} \\
\text{gen}_B3 &= \{ \text{d6} \} \\
\text{kill}_B3 &= \{ \text{d3} \} \\
\text{gen}_B4 &= \{ \text{d7} \} \\
\text{kill}_B4 &= \{ \text{d1, d4} \}
\end{align*}
\]
Extending transfer equations from statements to blocks

Composition of \( f_1 \) and \( f_2 \):

\[
\begin{align*}
f_1(x) &= \text{gen}_1 \cup (x - \text{kill}_1) \\
f_2(x) &= \text{gen}_2 \cup (x - \text{kill}_2) \\
f_2( f_1(x) ) &= \text{gen}_2 \cup ( (\text{gen}_1 \cup (x - \text{kill}_1)) - \text{kill}_2 ) \\
&= \text{gen}_2 \cup ( (\text{gen}_1 - \text{kill}_2) \cup ((x - \text{kill}_1) - \text{kill}_2)) \\
&= \text{gen}_2 \cup (\text{gen}_1 - \text{kill}_2) \cup (x - (\text{kill}_1 \cup \text{kill}_2))
\end{align*}
\]
Extending transfer equations from statements to blocks

In general:

\[ f_B(x) = \text{gen}_B \cup (x - \text{kill}_B) \]

\[ \text{kill}_B = \bigcup_{i \in n} \text{kill}_i \]

\[ \text{gen}_B = \text{gen}_n \cup (\text{gen}_{n-1} - \text{kill}_n) \cup (\text{gen}_{n-2} - \text{kill}_{n-1} - \text{kill}_n) \cup \ldots \cup (\text{gen}_1 - \text{kill}_2 - \text{kill}_3 - \ldots - \text{kill}_n) \]
Extending transfer equations from statements to blocks

“The gen set contains all the definitions inside the block that are "visible" immediately after the block - we refer to them as downwards exposed. A definition is downwards exposed in a basic block only if it is not "killed" by a subsequent definition to the same variable inside the same basic block.” [p. 605]
Iterative algorithm for reaching definitions

Algorithm [p. 606]

INPUT: A flow graph for which kill_B and gen_B have been computed for each block B.

OUTPUT: IN[B] and OUT[B], the set of definitions reaching the entry and exit of each block B of the flow graph

METHOD:

\[
\text{OUT}[\text{ENTRY}] = \emptyset
\]

for (each basic block B other than ENTRY) \{ \text{OUT}[B] = \emptyset \}

while (changes to any OUT occurs) \{

for (each basic block B other than ENTRY) \{

\[
\text{IN}[B] = \bigcup_{P \text{ a predecessor of } B} \text{OUT}[P]
\]

\[
\text{OUT}[B] = \text{gen}_B \cup (\text{IN}[B] - \text{kill}_B)
\]

\}

\}

See footnote 4 on page 606
Iterative algorithm for reaching definitions

Algorithm [p. 606]

INPUT: A flow graph for which kill\textsubscript{B} and gen\textsubscript{B} have been computed for each block B.

OUTPUT: IN[B] and OUT[B], the set of definitions reaching the entry and exit of each block B of the flow graph.

METHOD:

\begin{verbatim}
OUT[ENTRY] = ∅
for (each basic block B other than ENTRY) \{ OUT[B] = ∅ \}
while (changes to any OUT occurs) \{
  for (each basic block B other than ENTRY) \{
    IN[B] = \bigcup_{\text{a predecessor of B}} OUT[P]
    OUT[B] = gen\textsubscript{B} \cup (IN[B] - kill\textsubscript{B})
  \}
\}
\end{verbatim}

Written this way to allow different entry conditions for different data flow algorithms.
Figure 9.13 (p. 604)

\[ \text{gen}_{B1} = \{ d1, d2, d3 \} \]
\[ \text{kill}_{B1} = \{ d4, d5, d6, d7 \} \]

\[ \text{gen}_{B2} = \{ d4, d5 \} \]
\[ \text{kill}_{B2} = \{ d1, d2, d7 \} \]

\[ \text{gen}_{B3} = \{ d6 \} \]
\[ \text{kill}_{B3} = \{ d3 \} \]

\[ \text{gen}_{B4} = \{ d7 \} \]
\[ \text{kill}_{B4} = \{ d1, d4 \} \]
**Example 9.12 - building off figure 9.13**

\[
\text{OUT[ENTRY]} = \emptyset
\]

for (each basic block \( B \) other than ENTRY) \{ OUT[\( B \)] = \emptyset \}

while (changes to any OUT occurs) {
  for (each basic block \( B \) other than ENTRY) {  
    \( \text{IN}[B] = \cup P \) a predecessor of \( B \)  OUT[\( P \)]  
    \( \text{OUT}[B] = \text{gen}_B \cup (\text{IN}[B] - \text{kill}_B) \)  
  }
}

\[
\begin{array}{|l|l|l|l|l|}
\hline
\text{B1} & \text{OUT}[B]^0 & \text{IN}[B]^1 & \text{OUT}[B]^1 & \text{IN}[B]^2 & \text{OUT}[B]^2 \\
\hline
\text{represent } d_i \text{ as a bit vector, where each } d \text{ is a definition from 9.13} & \text{Union of sets } A \cup B: A \text{ OR } B & \text{Difference of sets } A - B: A \text{ AND } B' & \text{Compute in order } B1, B2, B3, B4, \text{EXIT} & \text{For example:} \\
\hline
\text{B2} & \text{IN}[B]^1 = \text{OUT}[B]^1 \cup \text{OUT}[B]^0 = 111 0000 \cup 000 0000 = 111 0000 & \text{OUT}[B]^1 = \text{gen}_B \cup (\text{IN}[B]^1 - \text{kill}_B) & \text{EXAMPLE:} \\
\hline
\text{B3} & \text{OUT}[B]^1 = 000 1100 + (111 0000 - 110 0001) = 001 1110 \\
\hline
\text{B4} & \text{EXIT} = 000 1100 + 001 0000 = 001 1100 \\
\hline
\end{array}
\]
Example 9.12 - building off figure 9.13

\[
\begin{align*}
\text{OUT}[\text{ENTRY}] &= \emptyset \\
\text{for (each basic block B other than ENTRY) } & \{ \text{ OUT}[B] = \emptyset \} \\
\text{while (changes to any OUT occurs) } & \{ \\
& \quad \text{for (each basic block B other than ENTRY) } \{ \\
& & \quad \text{IN}[B] = \cup \text{ a predecessor of B } \text{ OUT}[P] \\
& & \quad \text{OUT}[B] = \text{gen}_B \cup ( \text{IN}[B] - \text{kill}_B ) \\
& \quad \} \\
& \} \\
\end{align*}
\]

<table>
<thead>
<tr>
<th></th>
<th>OUT[B](^0)</th>
<th>IN[B](^1)</th>
<th>OUT[B](^1)</th>
<th>IN[B](^2)</th>
<th>OUT[B](^2)</th>
</tr>
</thead>
<tbody>
<tr>
<td>B1</td>
<td>000 0000</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>B2</td>
<td>000 0000</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>B3</td>
<td>000 0000</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>B4</td>
<td>000 0000</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>EXIT</td>
<td>000 0000</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Example 9.12

OUT[ENTRY] = ∅

for (each basic block B other than ENTRY) { OUT[B] = ∅ }

while (changes to any OUT occurs) {
    for (each basic block B other than ENTRY) {
        IN[B] = ∪ \text{a predecessor of B OUT[P]}
        OUT[B] = gen_{B} \cup (IN[B] - kill_{B})
    }
}

<table>
<thead>
<tr>
<th></th>
<th>OUT[B]^{0}</th>
<th>IN[B]^{1}</th>
<th>OUT[B]^{1}</th>
<th>IN[B]^{2}</th>
<th>OUT[B]^{2}</th>
</tr>
</thead>
<tbody>
<tr>
<td>B1</td>
<td>000 0000</td>
<td>000 0000</td>
<td>111 0000</td>
<td></td>
<td></td>
</tr>
<tr>
<td>B2</td>
<td>000 0000</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>B3</td>
<td>000 0000</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>B4</td>
<td>000 0000</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>EXIT</td>
<td>000 0000</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

IN[B1] = pred(B1) = ENTRY
OUT[B1] = gen_{B1} \cup (IN[B1] - kill_{B1})

\text{gen}_{B1} = \{d1, d2, d3\}
\text{kill}_{B1} = \{d4, d5, d6, d7\}
Example 9.12

\[ \text{OUT}[\text{ENTRY}] = \emptyset \]
for (each basic block \( B \) other than \( \text{ENTRY} \)) \{ \text{OUT}[B] = \emptyset \}
while (changes to any OUT occurs) \{
   for (each basic block \( B \) other than \( \text{ENTRY} \)) \{
      \text{IN}[B] = \bigcup \text{pred}(B) \cup \text{OUT}[P]
      \text{OUT}[B] = \text{gen}_B \cup (\text{IN}[B] - \text{kill}_B)
   \}
\}

<table>
<thead>
<tr>
<th>( B )</th>
<th>( \text{OUT}[B]^0 )</th>
<th>( \text{IN}[B]^1 )</th>
<th>( \text{OUT}[B]^1 )</th>
<th>( \text{IN}[B]^2 )</th>
<th>( \text{OUT}[B]^2 )</th>
</tr>
</thead>
<tbody>
<tr>
<td>( B_1 )</td>
<td>000 0000</td>
<td>000 0000</td>
<td>111 0000</td>
<td></td>
<td></td>
</tr>
<tr>
<td>( B_2 )</td>
<td>000 0000</td>
<td>111 0000</td>
<td>001 1100</td>
<td></td>
<td></td>
</tr>
<tr>
<td>( B_3 )</td>
<td>000 0000</td>
<td></td>
<td></td>
<td>( \text{IN}[B_2] = \text{pred}(B_2) = \text{OUT}[B_1] \cup \text{OUT}[B_4] )</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>( \text{OUT}[B_2] = \text{gen}<em>{B_2} \cup (\text{IN}[B_2] - \text{kill}</em>{B_2}) )</td>
<td></td>
</tr>
<tr>
<td>( B_4 )</td>
<td>000 0000</td>
<td></td>
<td></td>
<td>( \text{gen}_{B_2} = { d_4, d_5 } )</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>( \text{kill}_{B_2} = { d_1, d_2, d_7 } )</td>
<td></td>
</tr>
<tr>
<td>( \text{EXIT} )</td>
<td>000 0000</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Example 9.12

\[
\text{OUT[ENTRY]} = \emptyset
\]

for (each basic block B other than ENTRY) { \text{OUT[B]} = \emptyset }

while (changes to any OUT occurs) {
  for (each basic block B other than ENTRY) {
    \text{IN[B]} = \bigcup \text{pred}(B) \ \text{OUT[P]}
    \text{OUT[B]} = \text{gen}_B \cup ( \text{IN[B]} - \text{kill}_B )
  }
}

Table:

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>B1</td>
<td>000 0000</td>
<td>000 0000</td>
<td>111 0000</td>
<td></td>
<td></td>
</tr>
<tr>
<td>B2</td>
<td>000 0000</td>
<td>111 0000</td>
<td>001 1100</td>
<td></td>
<td></td>
</tr>
<tr>
<td>B3</td>
<td>000 0000</td>
<td>001 1100</td>
<td>000 1110</td>
<td></td>
<td></td>
</tr>
<tr>
<td>B4</td>
<td>000 0000</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>EXIT</td>
<td>000 0000</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Example 9.12

\[
\text{OUT}[\text{ENTRY}] = \emptyset
\]

for (each basic block \(B\) other than \(\text{ENTRY}\)) \{ \text{OUT}[B] = \emptyset \} 

while (changes to any OUT occurs) \{ 

for (each basic block \(B\) other than \(\text{ENTRY}\)) \{ 

\[
\text{IN}[B] = \bigcup \text{a predecessor of } B \text{ OUT}[P]
\]

\[
\text{OUT}[B] = \text{gen}_B \cup (\text{IN}[B] - \text{kill}_B)
\]

\}

\}

\[
d4: i = i + 1 \\
d5: j = j - 1 \\
d7: i = u3
\]

\[
\text{IN}[B4] = \{d7\} \\
\text{kill}_B = \{d1, d4\}
\]

\[
\text{gen}_B = \{d7\} \\
\text{kill}_B = \{d1, d4\}
\]

\[
\text{OUT}[B4] = \text{gen}_B \cup (\text{IN}[B4] - \text{kill}_B)
\]

\[
\text{OUT}[B2] = \bigcup \text{pred of } B2 \text{ OUT}[P]
\]

\[
\text{OUT}[B3] = \bigcup \text{pred of } B3 \text{ OUT}[P]
\]
Example 9.12

\[
\text{OUT}[\text{ENTRY}] = \emptyset
\]

for (each basic block B other than ENTRY) { \text{OUT}[B] = \emptyset }

while (changes to any OUT occurs) {
    for (each basic block B other than ENTRY) {
        \text{IN}[B] = \bigcup P \text{ a predecessor of } B \ \text{OUT}[P]
        \text{OUT}[B] = \text{gen}_B \cup ( \text{IN}[B] - \text{kill}_B )
    }
}

\begin{align*}
\text{IN}[\text{EXIT}] &= \text{OUT}[\text{B4}] \\
\text{OUT}[\text{EXIT}] &= \text{IN}[\text{EXIT}] 
\end{align*}

<table>
<thead>
<tr>
<th></th>
<th>\text{OUT}[B]^0</th>
<th>\text{IN}[B]^1</th>
<th>\text{OUT}[B]^1</th>
<th>\text{IN}[B]^2</th>
<th>\text{OUT}[B]^2</th>
</tr>
</thead>
<tbody>
<tr>
<td>B1</td>
<td>000 0000</td>
<td>000 0000</td>
<td>111 0000</td>
<td></td>
<td></td>
</tr>
<tr>
<td>B2</td>
<td>000 0000</td>
<td>111 0000</td>
<td>001 1100</td>
<td></td>
<td></td>
</tr>
<tr>
<td>B3</td>
<td>000 0000</td>
<td>001 1100</td>
<td>000 1110</td>
<td></td>
<td></td>
</tr>
<tr>
<td>B4</td>
<td>000 0000</td>
<td>001 1110</td>
<td>001 0111</td>
<td></td>
<td></td>
</tr>
<tr>
<td>EXIT</td>
<td>000 0000</td>
<td>001 0111</td>
<td>001 0111</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
\[\text{OUT[ENTRY]} = \emptyset\]

for (each basic block B other than ENTRY) \{ \text{OUT[B]} = \emptyset \}

while (changes to any \text{OUT} occurs) \{
  for (each basic block B other than ENTRY) \{
    \text{IN[B]} = \cup P \text{ a predecessor of } B \text{ OUT}[P]
    \text{OUT[B]} = \text{gen}\_B \cup (\text{IN[B]} - \text{kill}\_B)
  \}
\}

\begin{array}{|c|c|c|c|c|}
\hline
\text{B1} & \text{OUT[B]\text{\textdegree}} & \text{IN[B]\text{\textdegree}} & \text{OUT[B]\text{\textdegree}} & \text{IN[B]\text{\textdegree}} & \text{OUT[B]\text{\textdegree}} \\
\hline
000 0000 & 000 0000 & 111 0000 & 000 0000 & 111 0000 \\
\hline
\text{B2} & \text{OUT[B]\text{\textdegree}} & \text{IN[B]\text{\textdegree}} & \text{OUT[B]\text{\textdegree}} & \text{IN[B]\text{\textdegree}} & \text{OUT[B]\text{\textdegree}} \\
\hline
000 0000 & 111 0000 & 001 1100 & 001 1100 & 001 1100 \\
\hline
\text{B3} & \text{OUT[B]\text{\textdegree}} & \text{IN[B]\text{\textdegree}} & \text{OUT[B]\text{\textdegree}} & \text{IN[B]\text{\textdegree}} & \text{OUT[B]\text{\textdegree}} \\
\hline
000 0000 & 001 1100 & 000 1110 & 001 0111 & 001 0111 \\
\hline
\text{B4} & \text{OUT[B]\text{\textdegree}} & \text{IN[B]\text{\textdegree}} & \text{OUT[B]\text{\textdegree}} & \text{IN[B]\text{\textdegree}} & \text{OUT[B]\text{\textdegree}} \\
\hline
000 0000 & 001 1110 & 001 0111 & 001 0111 & 001 0111 \\
\hline
\end{array}
$$\text{OUT[ENTRY]} = \emptyset$$

for (each basic block \( B \) other than \( \text{ENTRY} \)) {
   \( \text{OUT}[B] = \emptyset \)
}

while (changes to any \( \text{OUT} \) occurs) {
   for (each basic block \( B \) other than \( \text{ENTRY} \)) {
      \( \text{IN}[B] = \cup \text{a predecessor of } B \text{ OUT}[P] \)
      \( \text{OUT}[B] = \text{gen}_B \cup (\text{IN}[B] - \text{kill}_B) \)
   }
}

\[
\begin{array}{cccccc}
\text{B1} & \text{OUT}[B]^0 & \text{IN}[B]^1 & \text{OUT}[B]^1 & \text{IN}[B]^2 & \text{OUT}[B]^2 \\
000 0000 & 000 0000 & 111 0000 & 000 0000 & 111 0000 \\
\text{B2} & 000 0000 & 111 0000 & 001 1100 & 111 0111 & 001 1110 \\
\text{B3} & 000 0000 & 001 1100 & 000 1110 \\
\text{B4} & 000 0000 & 001 1110 & 001 0111 \\
\text{EXIT} & 000 0000 & 001 0111 & 001 0111 \\
\end{array}
\]
\[ OUT[ENTRY] = \emptyset \]

for (each basic block B other than ENTRY) { \[ OUT[B] = \emptyset \] }

while (changes to any OUT occurs) {
    for (each basic block B other than ENTRY) {
        \[ IN[B] = \bigcup P \text{ a predecessor of } B \text{ OUT}[^P] \]
        \[ OUT[B] = \text{gen}_B \cup (IN[B] - \text{kill}_B) \]
    }
}

<table>
<thead>
<tr>
<th></th>
<th>\text{\textbf{OUT}[B]^0}</th>
<th>\text{\textbf{IN}[B]^1}</th>
<th>\text{\textbf{OUT}[B]^1}</th>
<th>\text{\textbf{IN}[B]^2}</th>
<th>\text{\textbf{OUT}[B]^2}</th>
</tr>
</thead>
<tbody>
<tr>
<td>B1</td>
<td>000 0000</td>
<td>000 0000</td>
<td>111 0000</td>
<td>000 0000</td>
<td>111 0000</td>
</tr>
<tr>
<td>B2</td>
<td>000 0000</td>
<td>111 0000</td>
<td>001 1100</td>
<td>111 0111</td>
<td>001 1110</td>
</tr>
<tr>
<td>B3</td>
<td>000 0000</td>
<td>001 1100</td>
<td>000 1110</td>
<td>001 1110</td>
<td>000 1110</td>
</tr>
<tr>
<td>B4</td>
<td>000 0000</td>
<td>001 1110</td>
<td>001 0111</td>
<td>001 0111</td>
<td></td>
</tr>
<tr>
<td>EXIT</td>
<td>000 0000</td>
<td>001 0111</td>
<td>001 0111</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
\[ \text{OUT}[\text{ENTRY}] = \emptyset \]

for (each basic block \( B \) other than \( \text{ENTRY} \)) { \( \text{OUT}[B] = \emptyset \) }

while (changes to any \( \text{OUT} \) occurs) {
    for (each basic block \( B \) other than \( \text{ENTRY} \)) {
        \( \text{IN}[B] = \bigcup \{ \text{a predecessor of } B \mid \text{OUT}[?] \} \)
        \( \text{OUT}[B] = \text{gen}_B \cup (\text{IN}[B] - \text{kill}_B) \)
    }
}

\begin{equation}
\begin{array}{cccccc}
\text{B1} & \text{OUT}^0 & \text{IN}^1 & \text{OUT}^1 & \text{IN}^2 & \text{OUT}^2 \\
000 & 0000 & 0000 & 0000 & 1110000 & 0000000 \\
000 & 0000 & 1110000 & 0011000 & 1110111 & 00111110 \\
000 & 0011000 & 0001110 & 0011100 & 0011110 & 0001110 \\
000 & 0011100 & 0010111 & 0011111 & 0011110 & 0010111 \\
000 & 0010111 & 0010111 & 0010111 & 0010111 & 0010111 \\
\end{array}
\end{equation}
$\text{OUT}[\text{ENTRY}] = \emptyset$

for (each basic block $B$ other than ENTRY) { $\text{OUT}[B] = \emptyset$ }

while (changes to any OUT occurs) {
  for (each basic block $B$ other than ENTRY) {
    $\text{IN}[B] = \cup P$ a predecessor of $B$ $\text{OUT}[P]$
    $\text{OUT}[B] = \text{gen}_B \cup (\text{IN}[B] - \text{kill}_B)$
  }
}

\begin{tabular}{|c|c|c|c|c|c|}
\hline

\hline
B1 & 000 0000 & 000 0000 & 111 0000 & 000 0000 & 111 0000 \\
\hline
B2 & 000 0000 & 111 0000 & 001 1100 & 111 0111 & 001 1110 \\
\hline
B3 & 000 0000 & 001 1100 & 000 1110 & 001 1110 & 000 1110 \\
\hline
B4 & 000 0000 & 001 1110 & 001 0111 & 001 1110 & 001 0111 \\
\hline
EXIT & 000 0000 & 001 0111 & 001 0111 & 001 0111 & 001 0111 \\
\hline
\end{tabular}
9.2.4 Reaching definitions

Useful for constant propagation and constant folding (§8.5.4 - p. 536, §9.4 - p. 632). Additional discussion and examples:

[en.wikipedia.org/wiki/Constant_folding](en.wikipedia.org/wiki/Constant_folding)

Useful for global common subexpression elimination (§9.1.4 - p. 588, §9.2.6 - p. 610, §9.5 - p. 639). Additional discussion and examples:

[en.wikipedia.org/wiki/Common_subexpression_elimination](en.wikipedia.org/wiki/Common_subexpression_elimination)
9.2.5 Live variable analysis

Useful for effective register management.

"After a value is computed in a register, and presumably used within a block, it is not necessary to store that value if it is dead at the end of the block. Also, if all registers are full and we need another register, we should favor using a register with a dead value, since that value does not have to be stored." [p. 608]
9.2.5 Live variable analysis

"In live variable analysis we wish to know for variable \( x \) and point \( p \) whether the value of \( x \) at \( p \) could be used along some path in the flow graph starting at \( p \). If so, we say \( x \) is live at \( p \); otherwise, \( x \) is dead at \( p \)." [p. 608]

In contrast to reaching analysis, which used a forward transfer function, live variable analysis uses a backward transfer function.
9.2.5 Live variable analysis

def \( B \) is "the set of variables defined in \( B \) prior to any use of that variable in \( B \)"

use \( B \) is "the set of variables whose values may be used in \( B \) prior to any definition of the variable"
9.2.5 Live variable analysis definitions, page 609

\[ \text{IN[EXIT]} = \emptyset \]

\[ \text{IN[B]} = \text{use}_B \cup (\text{OUT}[B] - \text{def}_B) \]

\[ \text{OUT}[B] = \bigcup_S \text{a successor of } B \quad \text{IN}[S] \]
9.2.5 Live variable analysis

Algorithm [p. 610]

INPUT: A flow graph with def and use computed for each block.

OUTPUT: IN[B] and OUT[B], the set of variables live on entry and exit of each block of the flow graph.

METHOD:

\[
\text{IN[EXIT]} = \emptyset
\]

for (each basic block B other than EXIT) \{ IN[B] = \emptyset \}

while (changes to any IN occur) \{
  for (each basic block B other than EXIT) \{
    \[ \text{OUT[B]} = \bigcup_{S \text{ a successor of } B} \text{IN}[S] \]
    \[ \text{IN[B]} = \text{use}_B \cup (\text{OUT[B]} - \text{def}_B) \]
  \}
\}
9.2.6 Available expressions

"An expression $x+y$ is available at a point $p$ if every path from the entry node to $p$ evaluates to $x+y$, and after the last such evaluation prior to reaching $p$, there are no subsequent assignments to $x$ or $y."$ [p. 610]
9.2.6 Available expressions

"...a block *kills* expression $x+y$ if it assigns (or may assign) $x$ or $y$ and does not subsequently recompute $x+y$." [p. 610]

"A block *generates* expression $x+y$ if it definitely evaluates $x+y$ and does not subsequently define $x$ or $y."$ [p. 611]
...the expression $4 \times i$ in block B3 will be a common subexpression if $4 \times i$ is available at the entry point of block B3.

[p 611]
Figure 9.17

\[ t_1 = 4 \times i \]

\[ t_2 = 4 \times i \]

"It will be available if i is not assigned a new value in block B2, ..." [p 611]

Here 4 \times i in B3 can be replaced by value of t1, regardless of which branch is taken.
Figure 9.17

"... or if ... 4 * i is recomputed after i is assigned in B2." [p 611]

Again, 4 * i in B3 can be replaced by value of t1, regardless of which branch is taken (since t1 contains the correct value of 4 * i in both cases)
Informally:

"If at point p set S of expressions is available, and q is the point after p, with statement \( x = y + z \) between them, then we form the set of expressions available at q by the following steps:

1. Add to S the expression \( y + z \).
2. Delete from S any expression involving variable x."

[p. 611]
### Example 9.15

<table>
<thead>
<tr>
<th>Statement</th>
<th>Available expressions</th>
</tr>
</thead>
<tbody>
<tr>
<td>$a = b + c$</td>
<td>$\emptyset$</td>
</tr>
<tr>
<td>$b = a - d$</td>
<td>${ a - d }$</td>
</tr>
<tr>
<td>$c = b + c$</td>
<td>${ a - d }$</td>
</tr>
<tr>
<td>$d = a - d$</td>
<td>$\emptyset$</td>
</tr>
</tbody>
</table>
9.2.6 Available expressions

"We can find available expressions in a manner reminiscent of the way reaching definitions are computed. Suppose U is the 'universal' set of all expressions appearing on the right of one or more statement of the program. For each block B, let $\text{IN}[B]$ be the set of expressions in U that are available at the point just before the beginning of B. Let $\text{OUT}[B]$ be the same for the point following the end of B. Define $\text{e_gen}_B$ to be the expressions generated by B and $\text{e_kill}_B$ to be the set of expressions in U killed in B. Note that IN, OUT, $\text{e_gen}$, and $\text{e_kill}$ can all be represented by bit vectors." [p. 612]
9.2.6 Available expressions definitions, page 612

\[
\begin{align*}
\text{OUT[ENTRY]} &= \emptyset \\
\text{OUT[B]} &= \text{e_gen}_B \cap (\text{IN[B]} - \text{e_kill}_B) \\
\text{IN[B]} &= \bigcap_{P \text{ a predecessor of } B} \text{OUT}[P]
\end{align*}
\]
9.2.6 Available expressions definitions, page 612

\[ \text{OUT[ENTRY]} = ∅ \]

\[ \text{OUT[B]} = e_{gen_B} \cap (\text{IN[B]} - e_{kill_B}) \]

\[ \text{IN[B]} = \bigcap_{P \text{ a predecessor of } B} \text{OUT[P]} \]

Note use of \( ∩ \) rather than \( ∪ \).

"...an expression is available at the beginning of a block only if it is available at the end of ALL its predecessors." [p. 612]
9.2.6 Available expressions

Algorithm [p. 614]

**INPUT:** A flow graph with $e_{\text{kill}}_B$ and $e_{\text{gen}}_B$ computed for each block $B$. The initial block is $B_1$.

**OUTPUT:** $\text{IN}[B]$ and $\text{OUT}[B]$, the set of expressions available at the entry and exit of each block of the flow graph.

**METHOD:**

\[
\text{OUT}[\text{ENTRY}] = \emptyset
\]

\[
\text{for (each basic block } B \text{ other than ENTRY) } \{ \text{OUT}[B] = \emptyset \}\]

\[
\text{while (changes to any OUT occur) } \{
\text{for (each basic block } B \text{ other than EXIT) } \{
\text{IN}[B] = \bigcap \{ \text{a predecessor of } B \} \text{ OUT}[_P] \\
\text{OUT}[B] = e_{\text{gen}}_B \cap \left( \text{IN}[B] - e_{\text{kill}}_B \right)
\}
\}
\]
9.2.6 Available expressions

Algorithm [p. 614]

INPUT: A flow graph with $e_{\text{kill}}_B$ and $e_{\text{gen}}_B$ computed for each block $B$. The initial block is $B_1$.

OUTPUT: $\text{IN}[B]$ and $\text{OUT}[B]$, the set of expressions available at the entry and exit of each block of the flow graph.

METHOD:

\[
\begin{align*}
\text{OUT}[\text{ENTRY}] &= \emptyset \\
\text{for} \ (\text{each basic block } B \text{ other than ENTRY}) \{ \text{OUT}[B] = U \} \\
\text{while} \ (\text{changes to any OUT occur}) \{ \\
\quad \text{for} \ (\text{each basic block } B \text{ other than EXIT}) \{ \\
\quad\quad \text{IN}[B] = \bigcap \ \text{P a predecessor of } B \ \text{OUT}[P] \\
\quad\quad \text{OUT}[B] = e_{\text{gen}}_B \cap (\text{IN}[B] - e_{\text{kill}}_B) \\
\quad \} \\
\} \\
\end{align*}
\]

Recall: $U$ is set of all expressions
### 9.2 Summary

<table>
<thead>
<tr>
<th></th>
<th>Reaching definitions</th>
<th>Live variables</th>
<th>Available expressions</th>
</tr>
</thead>
<tbody>
<tr>
<td>Domain</td>
<td>sets of definitions</td>
<td>sets of variables</td>
<td>sets of expressions</td>
</tr>
<tr>
<td>Direction</td>
<td>forward</td>
<td>backward</td>
<td>forward</td>
</tr>
<tr>
<td>Transfer function</td>
<td>$\text{gen}_B U (x - \text{kill}_B)$</td>
<td>$\text{use}_B U (x - \text{def}_B)$</td>
<td>$e_{\text{gen}<em>B} \cap (x - e</em>{\text{kill}_B})$</td>
</tr>
<tr>
<td>Boundary</td>
<td>$\text{OUT}[\text{ENTRY}] = \emptyset$</td>
<td>$\text{IN}[\text{EXIT}] = \emptyset$</td>
<td>$\text{OUT}[\text{ENTRY}] = \emptyset$</td>
</tr>
<tr>
<td>Meet ($\land$)</td>
<td>$U$</td>
<td>$U$</td>
<td>$\cap$</td>
</tr>
<tr>
<td>Equations</td>
<td>$\text{OUT}[B] = f_B(\text{IN}[B])$</td>
<td>$\text{IN}[B] = f_B(\text{OUT}[B])$</td>
<td>$\text{OUT}[B] = f_B(\text{IN}[B])$</td>
</tr>
<tr>
<td></td>
<td>$\text{IN}[B] = \land_{P,pred(B)} \text{OUT}[P]$</td>
<td>$\text{OUT}[B] = \land_{S,succ(B)} \text{IN}[S]$</td>
<td>$\text{IN}[B] = \land_{P,pred(B)} \text{OUT}[P]$</td>
</tr>
<tr>
<td>Initialize</td>
<td>$\text{OUT}[B] = \emptyset$</td>
<td>$\text{IN}[B] = \emptyset$</td>
<td>$\text{OUT}[B] = U$</td>
</tr>
</tbody>
</table>