Our complete lattice is given by:

$$\mathbb{B} = 2^{Expr \setminus Vars}$$

with $$\subseteq = \supseteq$$

The effect $$[k]^\#$$ of an edge $$k = (u, lab, v)$$ only depends on $$lab$$, i.e., $$[k]^\# = [lab]^\#$$ where:

$$[;]^\# B = B$$

$$[\text{Pos}(e)]^\# B = [\text{Neg}(e)]^\# B = B \cup \{e\}$$

$$[x = e;]^\# B = (B \setminus \text{Expr}_x) \cup \{e\}$$

$$[x = M[e];]^\# B = (B \setminus \text{Expr}_x) \cup \{e\}$$

$$[M[e_1] = e_2;]^\# B = B \cup \{e_1, e_2\}$$
These effects are all distributive. Thus, the least solution of the constraint system yields precisely the MOP — given that stop is reachable from every program point  :-)

Example:

\[
x = M[a]; \quad y_1 = x + 1; \\
y_2 = x + 1; \\
M[x] = y_1 + y_2;
\]
A point $u$ is called safe for $e$, if $e \in \mathcal{A}[u] \cup \mathcal{B}[u]$, i.e., $e$ is either available or very busy.

Idea:

- We insert computations of $e$ such that $e$ becomes available at all safe program points :-)
- We insert $T_e = e$; after every edge $(u, lab, v)$ with

$$e \in \mathcal{B}[v] \setminus [lab]^\mathcal{A}(\mathcal{A}[u] \cup \mathcal{B}[u])$$
Transformation 5.1:

\[ T_e = e; \quad (e \in B[v] \setminus \text{lab}_A (A[u] \cup B[u])) \]

\[ T_e = e; \quad (e \in B[v]) \]
Transformation 5.2:

\[ u \xrightarrow{x = e;} u \xrightarrow{x = T_e;} \]

// analogously for the other uses of \( e \)
// at old edges of the program.
Bernhard Steffen, Dortmund

Jens Knoop, Wien
In the Example:

$x = M[a];$

$y_1 = x + 1;$

$y_2 = x + 1;$

$M[x] = y_1 + y_2;$

<table>
<thead>
<tr>
<th></th>
<th>$A$</th>
<th>$B$</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>$\emptyset$</td>
<td>$\emptyset$</td>
</tr>
<tr>
<td>1</td>
<td>$\emptyset$</td>
<td>$\emptyset$</td>
</tr>
<tr>
<td>2</td>
<td>$\emptyset$</td>
<td>${x + 1}$</td>
</tr>
<tr>
<td>3</td>
<td>$\emptyset$</td>
<td>${x + 1}$</td>
</tr>
<tr>
<td>4</td>
<td>${x + 1}$</td>
<td>${x + 1}$</td>
</tr>
<tr>
<td>5</td>
<td>$\emptyset$</td>
<td>${x + 1}$</td>
</tr>
<tr>
<td>6</td>
<td>${x + 1}$</td>
<td>${y_1 + y_2}$</td>
</tr>
<tr>
<td>7</td>
<td>${x + 1}$</td>
<td>$\emptyset$</td>
</tr>
</tbody>
</table>
In the Example:

\[ x = M[a]; \]
\[ y_1 = x + 1; \]
\[ y_2 = x + 1; \]
\[ M[x] = y_1 + y_2; \]

<table>
<thead>
<tr>
<th></th>
<th>(A)</th>
<th>(B)</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>2</td>
<td>0</td>
<td>({x + 1})</td>
</tr>
<tr>
<td>3</td>
<td>0</td>
<td>({x + 1})</td>
</tr>
<tr>
<td>4</td>
<td>({x + 1})</td>
<td>({x + 1})</td>
</tr>
<tr>
<td>5</td>
<td>0</td>
<td>({x + 1})</td>
</tr>
<tr>
<td>6</td>
<td>({x + 1})</td>
<td>({y_1 + y_2})</td>
</tr>
<tr>
<td>7</td>
<td>({x + 1})</td>
<td>0</td>
</tr>
</tbody>
</table>
Im Example:

$T = x + 1$;

$x = M[a]$;

$y_1 = T$;

$T = x + 1$;

$y_2 = T$;

$M[x] = y_1 + y_2$;

<table>
<thead>
<tr>
<th></th>
<th>$A$</th>
<th>$B$</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>$\emptyset$</td>
<td>$\emptyset$</td>
</tr>
<tr>
<td>1</td>
<td>$\emptyset$</td>
<td>$\emptyset$</td>
</tr>
<tr>
<td>2</td>
<td>$\emptyset$</td>
<td>${x + 1}$</td>
</tr>
<tr>
<td>3</td>
<td>$\emptyset$</td>
<td>${x + 1}$</td>
</tr>
<tr>
<td>4</td>
<td>${x + 1}$</td>
<td>${x + 1}$</td>
</tr>
<tr>
<td>5</td>
<td>$\emptyset$</td>
<td>${x + 1}$</td>
</tr>
<tr>
<td>6</td>
<td>${x + 1}$</td>
<td>${y_1 + y_2}$</td>
</tr>
<tr>
<td>7</td>
<td>${x + 1}$</td>
<td>$\emptyset$</td>
</tr>
</tbody>
</table>
Correctness:

Let $\pi$ denote a path reaching $v$ after which a computation of an edge with $e$ follows.

Then there is a maximal suffix of $\pi$ such that for every edge $k = (u, lab, u')$ in the suffix:

$$e \in [lab]_A^\#(A[u] \cup B[u])$$
Correctness:

Let $\pi$ denote a path reaching $v$ after which a computation of an edge with $e$ follows.

Then there is a maximal suffix of $\pi$ such that for every edge $k = (u, lab, u')$ in the suffix:

$$e \in [lab]_A\{A[u] \cup B[u]\}$$

In particular, no variable in $e$ receives a new value :-)

Then $T_e = e;$ is inserted before the suffix :-))
We conclude:

- Whenever the value of \( e \) is required, \( e \) is available \( :) \)
  \[ \implies \text{correctness of the transformation} \]

- Every \( T = e \); which is inserted into a path corresponds to an \( e \) which is replaced with \( T \) \( :) \)
  \[ \implies \text{non-degradation of the efficiency} \]
1.8 Application: Loop-invariant Code

Example:

for \( (i = 0; i < n; i++) \)
\[ a[i] = b + 3; \]

// The expression \( b + 3 \) is recomputed in every iteration :-(
// This should be avoided :-)
The Control-flow Graph:

0
\[ i = 0; \]

1
\[ \text{Neg}(i < n) \]
\[ \text{Pos}(i < n) \]

7
\[ y = b + 3; \]

2

3
\[ A_1 = A + i; \]

4
\[ M[A_1] = y; \]

5
\[ i = i + 1; \]

6
Warning: \( T = b + 3; \) may not be placed before the loop:

\[
\begin{align*}
i &= 0; \\
T &= b + 3; \\
\neg (i < n) &\quad \text{Neg}(i < n) \\
\pos (i < n) &\quad \text{Pos}(i < n) \\
y &= T; \\
A_1 &= A + i; \\
M[A_1] &= y; \\
i &= i + 1;
\end{align*}
\]

\[\Rightarrow\] There is no decent place for \( T = b + 3; \) :-(

440
Idea: Transform into a do-while-loop ...

\[
i = 0;\\
i = i + 1;\\
A_1 = A + i;\\
M[A_1] = y;\\
y = b + 3;
\]
... now there is a place for $T = e; \quad :-)$

```

0

i = 0;

1

Pos(i < n)

T = b + 3;

2

Neg(i < n)

y = T;

3

A_1 = A + i;

4

M[A_1] = y;

5

i = i + 1;

6

Pos(i < n)

Neg(i < n)

7
```
Application of T5 (PRE):

\[ i = 0; \]
\[ y = b + 3; \]
\[ A_1 = A + i; \]
\[ M[A_1] = y; \]
\[ i = i + 1; \]

<table>
<thead>
<tr>
<th></th>
<th>( A )</th>
<th>( B )</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>( \emptyset )</td>
<td>( \emptyset )</td>
</tr>
<tr>
<td>1</td>
<td>( \emptyset )</td>
<td>( \emptyset )</td>
</tr>
<tr>
<td>2</td>
<td>( \emptyset )</td>
<td>{ ( b + 3 ) }</td>
</tr>
<tr>
<td>3</td>
<td>{ ( b + 3 ) }</td>
<td>( \emptyset )</td>
</tr>
<tr>
<td>4</td>
<td>{ ( b + 3 ) }</td>
<td>( \emptyset )</td>
</tr>
<tr>
<td>5</td>
<td>{ ( b + 3 ) }</td>
<td>( \emptyset )</td>
</tr>
<tr>
<td>6</td>
<td>{ ( b + 3 ) }</td>
<td>( \emptyset )</td>
</tr>
<tr>
<td>7</td>
<td>( \emptyset )</td>
<td>( \emptyset )</td>
</tr>
</tbody>
</table>
Application of T5 (PRE):

$\begin{align*}
i &= 0;\\
A_1 &= A + i;\\
M[A_1] &= y;\\
i &= i + 1;
\end{align*}$

<table>
<thead>
<tr>
<th></th>
<th>$\mathcal{A}$</th>
<th>$\mathcal{B}$</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>$\emptyset$</td>
<td>$\emptyset$</td>
</tr>
<tr>
<td>1</td>
<td>$\emptyset$</td>
<td>$\emptyset$</td>
</tr>
<tr>
<td>2</td>
<td>$\emptyset$</td>
<td>{ $b+3$ }</td>
</tr>
<tr>
<td>3</td>
<td>{ $b+3$ }</td>
<td>$\emptyset$</td>
</tr>
<tr>
<td>4</td>
<td>{ $b+3$ }</td>
<td>$\emptyset$</td>
</tr>
<tr>
<td>5</td>
<td>{ $b+3$ }</td>
<td>$\emptyset$</td>
</tr>
<tr>
<td>6</td>
<td>{ $b+3$ }</td>
<td>$\emptyset$</td>
</tr>
<tr>
<td>7</td>
<td>$\emptyset$</td>
<td>$\emptyset$</td>
</tr>
</tbody>
</table>
Conclusion:

- Elimination of partial redundancies may move loop-invariant code out of the loop :-(
- This only works properly for **do-while-loops** :-(
- To optimize other loops, we transform them into **do-while-loops** before-hand:

\[
\text{while (} b \text{) stmt} \quad \Longrightarrow \quad \text{if (} b \text{)} \\
\text{do stmt} \\
\text{while (} b \text{);} \\
\Longrightarrow \quad \text{Loop Rotation}
\]
Problem:

If we do not have the source program at hand, we must re-construct potential loop headers ;-)  

\[ u \text{ pre-dominates } v \], if every path \( \pi: \text{start} \rightarrow^* v \) contains \( u \). We write: \( u \Rightarrow v \).

“\( \Rightarrow \)” is reflexive, transitive and anti-symmetric :-)
Computation:

We collect the nodes along paths by means of the analysis:

\[ P = 2^{Nodes}, \quad \subseteq = \supseteq \]

\[ \llbracket (_-, -, v) \rrbracket ^{\#} P = P \cup \{ v \} \]

Then the set \( \mathcal{P}[v] \) of pre-dominators is given by:

\[ \mathcal{P}[v] = \bigcap \{ \llbracket \pi \rrbracket ^{\#} \{ start \} \mid \pi : start \rightarrow^* v \} \]
Since $[k]^\sharp$ are distributive, the $P[v]$ can computed by means of fixpoint iteration :-)

Example:

\[
\begin{array}{c|c}
\hline
& P \\
\hline
0 & \{0\} \\
1 & \{0, 1\} \\
2 & \{0, 1, 2\} \\
3 & \{0, 1, 2, 3\} \\
4 & \{0, 1, 2, 3, 4\} \\
5 & \{0, 1, 5\} \\
\hline
\end{array}
\]
The partial ordering “⇒” in the example:

\[
\begin{array}{c|c}
 & \mathcal{P} \\
0 & \{0\} \\
1 & \{0, 1\} \\
2 & \{0, 1, 2\} \\
3 & \{0, 1, 2, 3\} \\
4 & \{0, 1, 2, 3, 4\} \\
5 & \{0, 1, 5\} \\
\end{array}
\]
Apparently, the result is a tree  :-) 

In fact, we have:

**Theorem:**

Every node $v$ has at most one immediate pre-dominator.

**Proof:**

Assume:

there are $u_1 \neq u_2$ which immediately pre-dominate $v$.

If $u_1 \Rightarrow u_2$ then $u_1$ not immediate.

Consequently, $u_1, u_2$ are incomparable  :-)
Now for every $\pi : \text{start} \rightarrow^* v$:

$$\pi = \pi_1 \pi_2 \quad \text{with} \quad \pi_1 : \text{start} \rightarrow^* u_1$$

$$\pi_2 : u_1 \rightarrow^* v$$

If, however, $u_1, u_2$ are incomparable, then there is path: $\text{start} \rightarrow^* v$

avoiding $u_2$:
Now for every $\pi : start \rightarrow^* v$:

$$\pi = \pi_1 \pi_2 \quad \text{with} \quad \pi_1 : start \rightarrow^* u_1$$

$$\pi_2 : u_1 \rightarrow^* v$$

If, however, $u_1, u_2$ are incomparable, then there is path: $start \rightarrow^* v$

avoiding $u_2$:
Observation:

The loop head of a while-loop pre-dominates every node in the body.

A back edge from the exit $u$ to the loop head $v$ can be identified through

$$v \in \mathcal{P}[u]$$

Accordingly, we define:
Transformation 6:

\[ v \]

\[ u \]

\[ u_1 \]

\[ u_2 \]

\[ u \]

\[ \text{Neg} \ (e) \]

\[ \text{Pos} \ (e) \]

\[ \text{lab} \]

\[ u_1 \not\in \mathcal{P}[u] \]

\[ u_2, v \in \mathcal{P}[u] \]

\[ v \]

\[ u_1 \]

\[ u_2 \]

\[ u \]

\[ \text{Neg} \ (e) \]

\[ \text{Pos} \ (e) \]

\[ \text{Neg} \ (e) \]

\[ \text{Pos} \ (e) \]

\[ \text{lab} \]

We duplicate the entry check to all back edges. :-)
... in the Example:

$i = 0;$

$\text{Neg}(i < n)$

$y = b + 3;$

$A_1 = A + i;$

$M[A_1] = y;$

$i = i + 1;$
... in the Example:

\[ \begin{align*}
    i &= 0; \\
    \text{Neg}(i < n) &\quad \text{Pos}(i < n) \\
    0, 1, 7 &\quad 0, 1, 2 \\
    y &= b + 3; \\
    A_1 &= A + i; \\
    M[A_1] &= y; \\
    i &= i + 1; \\
    0, 1, 7 &\quad 0, 1, 2, 3, 4, 5, 6
\end{align*} \]
... in the Example:

0, 1, 7

\( i = 0; \)

\( \text{Neg}(i < n) \)

\( \text{Pos}(i < n) \)

\( y = b + 3; \)

\( A_1 = A + i; \)

\( M[A_1] = y; \)

\( i = i + 1; \)

0, 1, 2, 3, 4, 5, 6

0, 1, 2, 3, 4, 5
... in the Example:

\[ i = 0; \]

Neg\((i < n)\) \quad Pos\((i < n)\)

0, 1

7

y = b + 3;

\[ A_1 = A + i; \]

\[ M[A_1] = y; \]

\[ i = i + 1; \]

Neg\((i < n)\) \quad Pos\((i < n)\)

0, 1, 2, 3

0, 1, 2, 3, 4

0, 1, 2, 3, 4, 5

0, 1, 2, 3, 4, 5, 6

458
Warning:

There are unusual loops which cannot be rotated:

Pre-dominators:
... but also **common ones** which cannot be rotated:

Here, the complete block between back edge and conditional jump should be duplicated  :-(
... but also common ones which cannot be rotated:

Here, the complete block between back edge and conditional jump should be duplicated  :-(

461