Problem Sheet 7: Program Slicing and Abstract Interpretation
Sample Solutions

Chris Poskitt*
ETH Zürich

Starred exercises (*) are more challenging than the others.

1 Program Slicing

i. Here is the program dependence graph for the program fragment (blue arrows are from the use-definition analysis; red arrows indicate control dependencies):

![Program Dependence Graph]

ii. For slicing criterion print(x), i.e. block 12, we get:

```plaintext
x := 0;
i := n;
while i > 0 do
    x := x + 1;
i := i - 1;
end
print(x);
```

*These solutions are adapted from previous iterations of the course when Stephan van Staden was the teaching assistant.
For slicing criterion print(y), i.e. block 13, we get:

```plaintext
y := 0;
i := n;
while i > 0 do
    i := i - 1;
j := i;
    while j > 0 do
        y := y + 1;
j := j - 1;
end
end
print(y);
```

2 Abstract Interpretation

i. We begin by mapping every variable to ⊥ (except for x, y in A₁, which are respectively mapped to +, ⊤ by assumption). Then, we iteratively update the (abstract) values of variables by applying the system of equations.

<table>
<thead>
<tr>
<th>Abstract States</th>
<th>Iterations →</th>
<th>Final Values</th>
</tr>
</thead>
<tbody>
<tr>
<td>A₁(x)</td>
<td>+</td>
<td>+</td>
</tr>
<tr>
<td>A₁(y)</td>
<td>✁</td>
<td>✁</td>
</tr>
<tr>
<td>A₂(x)</td>
<td>⊥ +</td>
<td>✁ ✁</td>
</tr>
<tr>
<td>A₂(y)</td>
<td>⊥ +</td>
<td>✁ ✁</td>
</tr>
<tr>
<td>A₃(x)</td>
<td>⊥ +</td>
<td>✁ ✁</td>
</tr>
<tr>
<td>A₃(y)</td>
<td>⊥ +</td>
<td>✁ ✁</td>
</tr>
<tr>
<td>A₄(x)</td>
<td>⊥ +</td>
<td>✁ ✁</td>
</tr>
<tr>
<td>A₄(y)</td>
<td>⊥ +</td>
<td>✁ ✁</td>
</tr>
<tr>
<td>A₅(x)</td>
<td>⊥ ⊥ 0</td>
<td>⊥ 0 0</td>
</tr>
<tr>
<td>A₅(y)</td>
<td>⊥ +</td>
<td>✁ ✁</td>
</tr>
</tbody>
</table>

ii. The analysis is not very precise: it cannot prove that y is positive when the program fragment completes (i.e. at A₅).

iii. (a) If we compute the factorial using a program that does not utilise the subtraction operator, then the result of the analysis becomes more precise:
Once we eliminate the problematic minus operator, the analysis becomes more precise:

\[
\begin{align*}
A_0(x) &= + \\
A_0(y) &= T \\
A_0(i) &= T \\
A_1(x) &= + \\
A_1(y) &= + \\
A_1(i) &= T \\
A_2(x) &= + \\
A_2(y) &= + \\
A_2(i) &= + \\
A_3(x) &= + \\
A_3(y) &= + \\
A_3(i) &= + \\
A_4(x) &= + \\
A_4(y) &= + \\
A_4(i) &= + \\
A_5(x) &= + \\
A_5(y) &= + \\
A_5(i) &= +
\end{align*}
\]

\(A_0 = [x \mapsto +, y \mapsto T, i \mapsto T]\)
\(A_1 = A_0[y \mapsto +]\)
\(A_2 = A_1[i \mapsto +] \cup A_2[i \mapsto A_4(i) \oplus +]\)
\(A_3 = A_2\)
\(A_4 = A_3[y \mapsto A_3(i) \otimes A_3(y)]\)
\(A_5 = A_2\)

(b) Perhaps changing the program for the analysis to work more precisely is not the best approach—let’s try to improve the analysis! We’ll try a so-called relational analysis with domain \(\mathcal{P}(\{-,0,0\} \times \{-,0,0\})\) to represent program states \((x, y)\). A relational analysis is more precise because the domain can express dependencies, or relationships, between \(x\) and \(y\).

We use the original version of the program fragment, but the new system of equations below:
We use the domain $\mathcal{P}(\{-,0,+\} \times \{-,0,+\})$ to represent the program state $(x, y)$. This is a so-called relational analysis. The relational analysis is more precise because the domain can express dependencies, or relationships, between $x$ and $y$.

\[
\begin{align*}
A_1 &= \{(+,\cdot), (+,0), (+,+)\} \\
A_2 &= \{(x,+) \mid (x,y) \in A_1 \} \cup \{(x,y') \mid (x',y') \in A_4 \text{ and } x \in x' \ominus +\} \\
A_3 &= A_2 \cap \{(x,y) \mid x \in \{-,0,+\} \text{ and } y \in \{-,0,\cdot\}\} \\
A_4 &= \{(x',y) \mid (x',y') \in A_3 \text{ and } y \in x' \otimes y'\} \\
A_5 &= A_2 \cap \{(0,y) \mid y \in \{-,0,+\}\}
\end{align*}
\]

and obtain a more precise analysis allowing us to deduce that $y$ will be positive after execution finishes:

<table>
<thead>
<tr>
<th>Iterations</th>
<th>Answer</th>
</tr>
</thead>
<tbody>
<tr>
<td>$A_1$</td>
<td>${(+,\cdot), (+,0), (+,+)}$</td>
</tr>
<tr>
<td>$A_2$</td>
<td>$\emptyset$</td>
</tr>
<tr>
<td>$A_3$</td>
<td>$\emptyset$</td>
</tr>
<tr>
<td>$A_4$</td>
<td>$\emptyset$</td>
</tr>
<tr>
<td>$A_5$</td>
<td>$\emptyset$</td>
</tr>
</tbody>
</table>
