Try it out 1.4

$$\mathcal{A}[\![A[x]/x]\!]\sigma = \mathcal{A}[\![A[x]]\!]\sigma \,/\, \mathcal{A}[\![x]\!]\sigma \xrightarrow[\sigma_A(A)=[7,9,13,17],\ 0\le 2<4]{\mathcal{A}[\![x]\!]\sigma=\sigma_v(x)=2} \quad v_0, v_1, v_2, v_3$$

$$= \mathcal{A}[\![13]\!]\sigma / 2 = 13/2 = 6.5$$

if $\sigma_v(x) = 4$:

$$\mathcal{A}[\![A[x]/x]\!]\sigma = \mathcal{A}[\![A[x]]\!]\sigma \,/\, \mathcal{A}[\![x]\!]\sigma \xrightarrow[\sigma_A(A)=[7,9,13,17],\ 0\le 4\not<4]{\mathcal{A}[\![x]\!]\sigma=\sigma_v(x)=4} \quad v_0, v_1, v_2, v_3$$

$$= \text{undefined}$$

if $\sigma_v(x) = 0$:

$$\mathcal{A}[\![A[x]/x]\!]\sigma = \mathcal{A}[\![A[x]]\!]\sigma \,/\, \mathcal{A}[\![x]\!]\sigma \xrightarrow[\sigma_A(A)=[7,9,13,17],\ 0\le 0<4]{\mathcal{A}[\![x]\!]\sigma=\sigma_v(x)=0} \quad v_0, v_1, v_2, v_3$$

$$\mathcal{A}[\![7]\!]\sigma / \sigma_v(x) = 7/0 \longrightarrow \text{undefined}$$

Try it out 1.5:

$$\mathcal{B}[\![(y>0)\wedge(x/y>0)]\!]\sigma = \mathcal{B}[\![y>0]\!]\sigma \wedge \mathcal{B}[\![x/y>0]\!]\sigma$$

$$= \left(\mathcal{A}[\![y]\!]\sigma > \mathcal{A}[\![0]\!]\sigma\right) \wedge \left(\mathcal{A}[\![x/y]\!]\sigma > \mathcal{A}[\![0]\!]\sigma\right)$$

$$= \left(\sigma_v(y) > 0\right) \wedge \left[\left(\mathcal{A}[\![x]\!]\sigma / \mathcal{A}[\![y]\!]\sigma\right) > 0\right] \xrightarrow{\sigma_v(x)=5,\ \sigma_v(y)=3}$$

$$= (3>0) \wedge \left(\frac{5}{3}>0\right) = \text{true} \wedge \text{true} = \text{true}$$

if $\sigma_v(y) = 0$:

$$\mathcal{B}[\![(y>0) \wedge (x/y>0)]\!]\sigma = \mathcal{B}[\![y>0]\!]\sigma \wedge \mathcal{B}[\![x/y>0]\!]\sigma$$

$$= \left(\mathcal{A}[\![y]\!]\sigma > \mathcal{A}[\![0]\!]\sigma\right) \wedge \left(\mathcal{A}[\![x/y]\!]\sigma > \mathcal{A}[\![0]\!]\sigma\right)$$

$$= \left(\sigma_v(y) > 0\right) \wedge \left[\left(\mathcal{A}[\![x]\!]\sigma / \mathcal{A}[\![y]\!]\sigma\right) > 0\right] \xRightarrow{\sigma_v(x)=5,\ \sigma_v(y)=0}$$

$$= (0 > 0) \wedge \left(\frac{5}{0} > 0\right) \Rightarrow \text{undefined}$$

Exercise 1.6:

Extension:

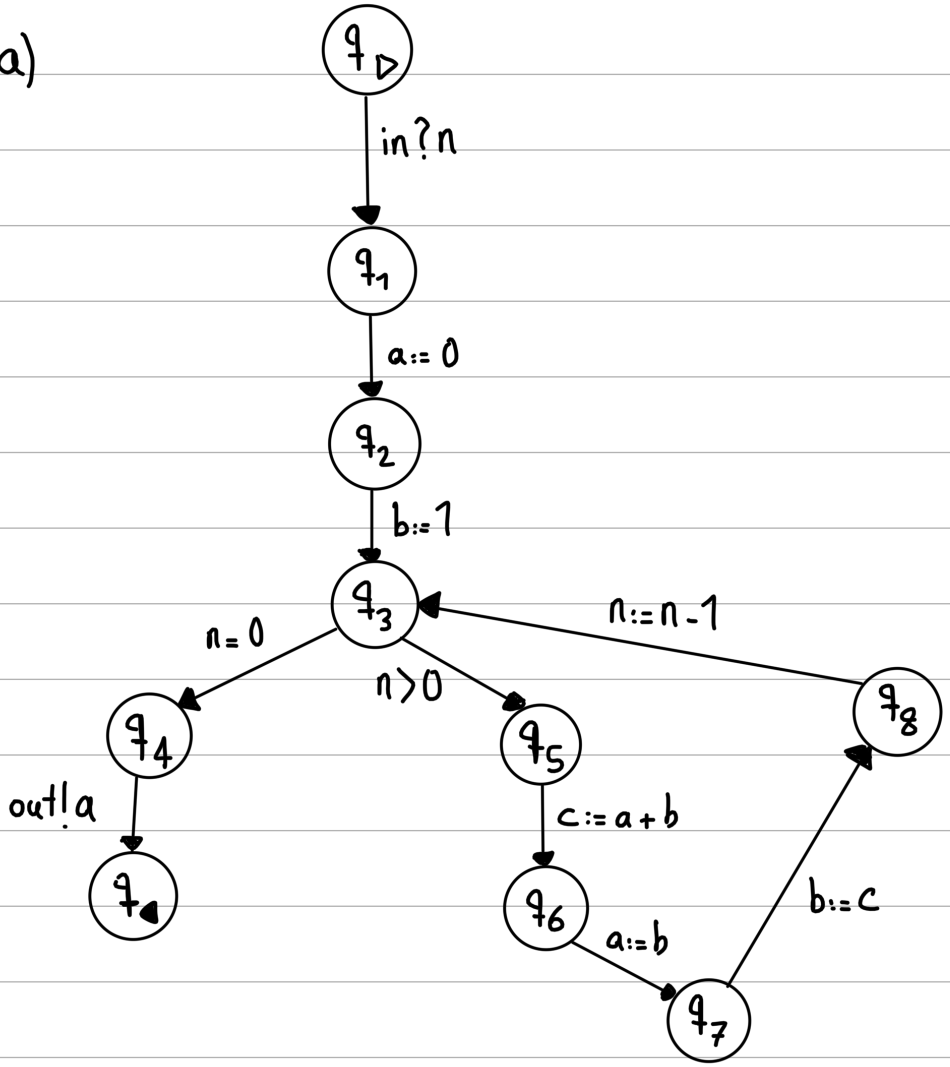
$$\mathcal{B}[\![b_1 \,\&\&\, b_2]\!]\sigma = \begin{cases} \text{false} & \text{if } \mathcal{B}[\![b_1]\!]\sigma = \text{false} \\ \text{false} & \text{if } \mathcal{B}[\![b_2]\!]\sigma = \text{false} \\ \text{true} & \text{if } \mathcal{B}[\![b_1]\!]\sigma = \mathcal{B}[\![b_2]\!]\sigma = \text{true} \\ \text{undefined} & \text{O.W} \end{cases}$$

$$\mathcal{B}[\![b_1 \,||\, b_2]\!]\sigma = \begin{cases} \text{true} & \text{if } \mathcal{B}[\![b_1]\!]\sigma = \text{true} \\ \text{true} & \text{if } \mathcal{B}[\![b_2]\!]\sigma = \text{true} \\ \text{false} & \text{if } \mathcal{B}[\![b_1]\!]\sigma = \mathcal{B}[\![b_2]\!]\sigma = \text{false} \\ \text{undefined} & \text{O.W} \end{cases}$$

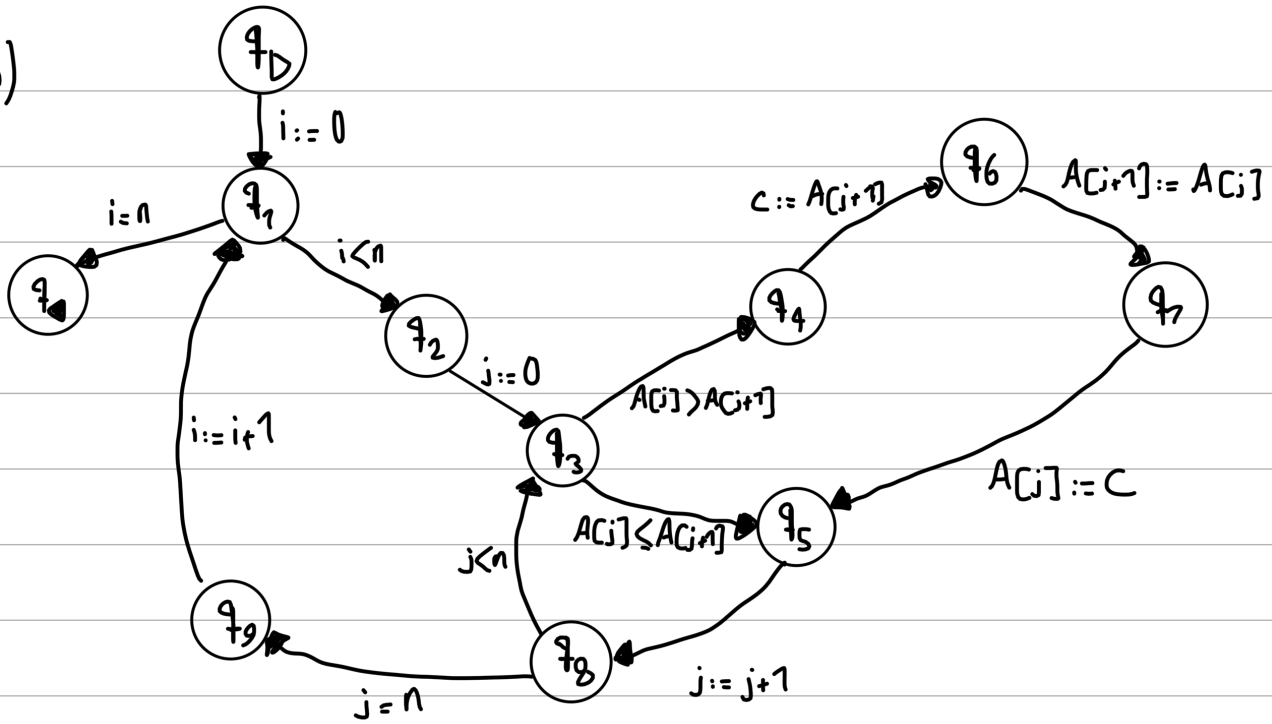
$\beta[\![(y>0) \,\&\&\, (x/y>0)]\!]\sigma = \text{false} \,\&\&\, \beta[\![x/y>0)]\!]\sigma = \text{false}$

Exercise 1.12:

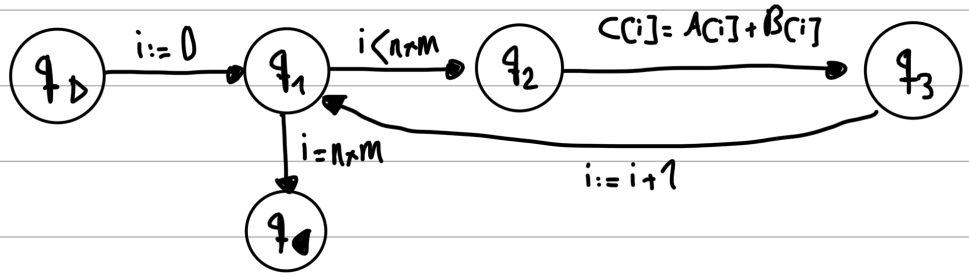
a)



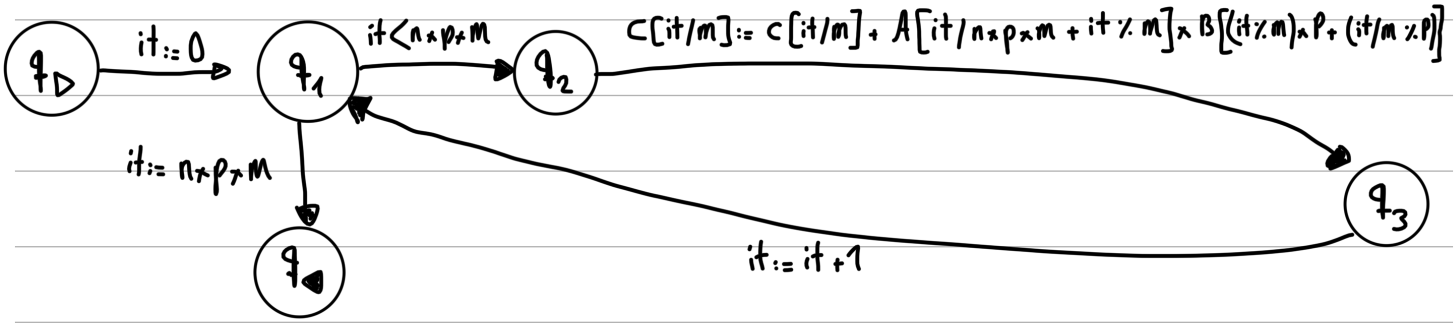
b)


q_0
i := 0
q_1
i = n
q_4
i < n
q_2
j := 0
q_3
i := i+1
q_9
j = n
q_8
A[j] ≤ A[j+1]
q_5
j < n
A[j] > A[j+1]
q_4
c := A[j+1]
q_6
A[j+1] := A[j]
q_7
A[j] := c
j := j+1

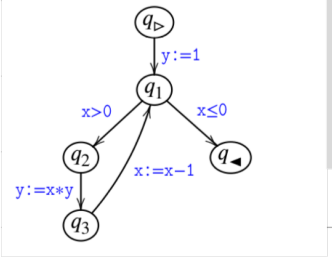
c)


q_0
i := 0
q_1
i < n*m
q_2
C[i] = A[i] + B[i]
q_3
i := i+1
i = n*m
q_4

d)


q_0
it := 0
q_1
it < n*p*m
q_2
C[it/m] := c[it/m] + A[it/n*p*m + it % m] * B[(it % m)*P + (it/m % P)]
q_3
it := it+1
it := n*p*m
q_4

Try it out 2.4:



(a) $q_\triangleright$

(b) $q_\triangleright, y := 1, q_1, x \leq 0, q_\blacktriangleleft$

(c) $q_1, x > 0, q_2, y := x * y, q_3$

(d) $q_\triangleright, y := 1, q_1, x > 0, q_2, y := x * y, q_3, x := x - 1, q_1, x > 0, q_2$

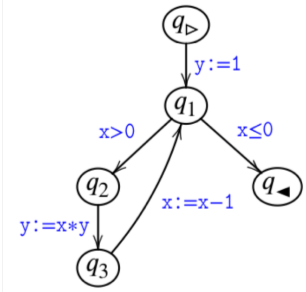
$Def(\pi_a) = \{(x, ?, q_\triangleright), (y, ?, q_\triangleright)\}$

$Def(\pi_b) = \{(x, ?, q_\triangleright), (y, q_\triangleright, q_1)\}$

$Def(\pi_c) = \{(x, ?, q_1), (y, q_2, q_3)\}$

$Def(\pi_d) = \{(x, q_3, q_1), (y, q_2, q_3)\}$

Try it out 2.6:



$RD(q_\triangleright) \supseteq \{(x, ?^1, q_\triangleright), (y, ?^2, q_\triangleright)\}$ $\pi = \{q_\triangleright\}^{1,2}$

$RD(q_1) \supseteq \{(x, ?^1, q_\triangleright), (x, q_3^2, q_1), (y, q_\triangleright^3, q_1), (y, q_2^4, q_3)\}$, $\pi = \{(q_\triangleright, q_1)^{1,3}, (q_\triangleright, q_1, q_2, q_3, q_1)^{2,4}\}$

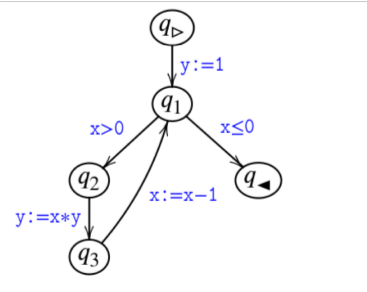
$RD(q_2) \supseteq \{(x, ?^1, q_\triangleright), (x, q_3^2, q_1), (y, q_\triangleright^3, q_1), (y, q_2^4, q_3)\}$, $\pi = \{(q_\triangleright, q_1, q_2)^{1,3}, (q_\triangleright, q_1, q_2, q_3, q_1, q_2)^{2,4}\}$

$RD(q_3) \supseteq \{(x, ?^1, q_\triangleright), (x, q_3^2, q_1), (y, q_2^3, q_3)\}$, $\pi = \{(q_\triangleright, q_1, q_2, q_3)^{1,3}, (q_\triangleright, q_1, q_2, q_3, q_1, q_2, q_3)^{2,3}\}$

$RD(q_\blacktriangleleft) \supseteq \{(x, ?^1, q_\triangleright), (x, q_3^2, q_1), (y, q_\triangleright^3, q_1), (y, q_2^4, q_3)\}$

$\pi = \{(q_\triangleright, q_1, q_\blacktriangleleft)^{1,3}, (q_\triangleright, q_1, q_2, q_3, q_1, q_\blacktriangleleft)^{2,4}\}$

Try it out 2.7:



$kill_{RD}(q_1, x>0, q_2) = gen_{RD}(q_1, x>0, q_2) = \emptyset$

$kill_{RD}(q_2, y:=x*y, q_3) = \{(y, q, q') \mid q \in Q_?, q' \in Q\}$

$gen_{RD}(q_2, y:=x*y, q_3) = \{(y, q_2, q_3)\}$

$kill_{RD}(q_3, x:=x-1, q_1) = \{(x, q, q') \mid q \in Q_?, q' \in Q\}$

$gen_{RD}(q_3, x:=x-1, q_1) = \{(x, q_3, q_1)\}$

$kill_{RD}(q_1, x\leq 0, q_\blacktriangleleft) = gen_{RD}(q_1, x\leq 0, q_\blacktriangleleft) = \emptyset$

Exercise 2.8.

Edge	$kill_{RD}$	gen_{RD}
$(q_0, in?x, q_1)$	$\{x\} \times Q? \times Q$	$\{(x, q_0, q_1)\}$
$(q_1, x \geq 0, q_2)$	$\{\}$	$\{\}$
$(q_2, i := 0, q_3)$	$\{i\} \times Q? \times Q$	$\{(i, q_2, q_3)\}$
$(q_3, y := -1, q_4)$	$\{y\} \times Q? \times Q$	$\{(y, q_3, q_4)\}$
$(q_4, i < n, q_6)$	$\{\}$	$\{\}$
$(q_6, A[i] \neq x, q_{10})$	$\{\}$	$\{\}$
$(q_{10}, i := i+1, q_4)$	$\{i\} \times Q? \times Q$	$\{(i, q_{10}, q_4)\}$
$(q_6, A[i] = x, q_7)$	$\{\}$	$\{\}$
$(q_7, B[i] := B[i] + B[i]/10, q_8)$	$\{\}$	$\{(B, q_7, q_8)\}$
$(q_8, y := B[i], q_9)$	$\{y\} \times Q? \times Q$	$\{(y, q_8, q_9)\}$
$(q_9, skip, q_5)$	$\{\}$	$\{\}$
$(q_4, \neg(i < n), q_5)$	$\{\}$	$\{\}$
$(q_5, out!y, q_{11})$	$\{\}$	$\{\}$
$(q_{11}, in?x, q_1)$	$\{x\} \times Q? \times Q$	$\{(x, q_{11}, q_1)\}$
$(q_1, x < 0, q_e)$	$\{\}$	$\{\}$

Exercise 2.9.

Yes, you can use the same approach for varibles and arrays in this setting (For both $kill_{RD}$ sets and Def)

Try it out 2.12.

$$RD(q_0) = \{(x,?,q_0),(y,?,q_0)\} \supseteq \{(x,?,q_0),(y,?,q_0)\}$$

$$RD(q_1) = \{(x,?,q_0),(x,q_3,q_1),(y,q_0,q_1),(y,q_2,q_3)\} \supseteq \{(x,?,q_0),(y,q_0,q_1)\}$$

$$RD(q_1) = \{(x,?,q_0),(x,q_3,q_1),(y,q_0,q_1),(y,q_2,q_3)\} \supseteq \{(x,q_3,q_1),(y,q_2,q_3)\}$$

$$RD(q_2) = \{(x,?,q_0),(x,q_3,q_1),(y,q_0,q_1),(y,q_2,q_3)\} \supseteq RD(q_1)$$

$$RD(q_3) = \{(x,?,q_0),(x,q_3,q_1),(y,q_2,q_3)\} \supseteq \{(x,?,q_0),(x,q_3,q_1),(y,q_2,q_3)\}$$

$$RD(q_4) = \{(x,?,q_0),(x,q_3,q_1),(y,q_0,q_1),(y,q_2,q_3)\} \supseteq RD(q_1)$$

Exercise 2.13:

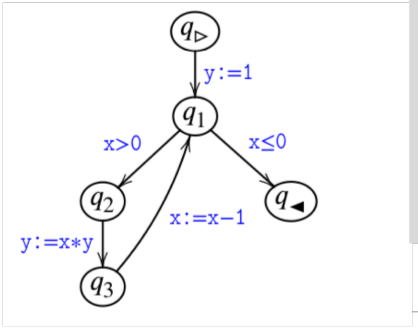
$$Def(A[a_1] := a_2) = \{A[a_1]\}, \quad Def(c?A[a]) = \{A[a]\}$$

$$Def(\pi, A[k]) = \begin{cases} (A[k], q_{i-1}, q_i) & \text{if } A[k] \in Def(a_i),\ \forall j>i:\ A[k] \notin Def(a_j) \\ (A[k], ?, q_0) & \text{if } \forall j:\ A[k] \in Def(a_j) \end{cases}$$

a	$kill_{RD}(q_0, a, q_4)$	$gen_{RD}(q_3, a, q_4)$
$A[a_1] := a_2$	$\{A[a_1]\} \times Q? \times Q$	$\{(A[a_1], q_0, q_4)\}$
$c?A[a]$	$\{A[a]\} \times Q? \times Q$	$\{(A[a], q_0, q_4)\}$

This redifinition makes compution more complex because we must pay attention to each entry of array; so the approximation can help us with this problem.

Try it out 2.14:



$$
\begin{aligned}
RD(q_\triangleright) &\supseteq \{(x,?,q_\triangleright),(y,?,q_\triangleright)\}\\
RD(q_1) &\supseteq \{(x,?,q_\triangleright),(x,q_3,q_1),(y,q_\triangleright,q_1),(y,q_2,q_3)\}\\
RD(q_2) &\supseteq \{(x,?,q_\triangleright),(x,q_3,q_1),(y,q_\triangleright,q_1),(y,q_2,q_3)\}\\
RD(q_3) &\supseteq \{(x,?,q_\triangleright),(x,q_3,q_1),(y,q_2,q_3)\}\\
RD(q_\blacktriangleleft) &\supseteq \{(x,?,q_\triangleright),(x,q_3,q_1),(y,q_\triangleright,q_1),(y,q_2,q_3)\}
\end{aligned}
$$

$\forall q \neq q_\triangleright,\ RD(q) = \emptyset,\quad RD(q_\triangleright) = \{(x,!,q_\triangleright),(y,!,q_\triangleright)\}$

1) $q_\triangleright, q_1$: $RD(q_1) = \{(x,?,q_\triangleright),(y,q_\triangleright,q_1)\}$

2) q_1, q_2: $RD(q_2) = \{(x,?,q_\triangleright),(y,q_\triangleright,q_1)\}$

3) q_2, q_3: $RD(q_3) = \{(x,?,q_\triangleright),(y,q_2,q_3)\}$

4) q_3, q_1: $RD(q_1) = \{(x,?,q_\triangleright),(x,q_3,q_1),(y,q_\triangleright,q_1),(y,q_2,q_3)\}$

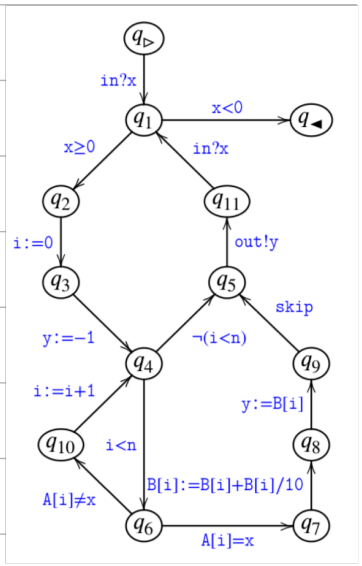
5) q_1, q_2: $RD(q_2) = \{(x,?,q_\triangleright),(x,q_3,q_1),(y,q_2,q_3),(y,q_\triangleright,q_1)\}$

6) q_2, q_3: $RD(q_3) = \{(x,?,q_\triangleright),(x,q_3,q_1),(y,q_2,q_3)\}$

7) q_1, q_4: $RD(q_4) = \{(x, ?, q_0), (x, q_3, q_1), (y, q_0, q_1), (y, q_2, q_3)\}$

End of algorithm

Exercise 2.15:




```
RD[0]: {('B', '?', 0), ('i', '?', 0), ('y', '?', 0), ('x', '?', 0)}
RD[1]: {('y', '?', 0), ('i', '10', '4'), ('y', '3', '4'), ('x', '0', '1'), ('B', '7', '8'), ('B', '?', 0), ('i', '?', 0), ('x', '11', '1'), ('y', '8', '9'), ('x', '?', 0), ('i', '2', '3')}
RD[2]: {('i', '10', '4'), ('x', '0', '1'), ('i', '?', 0), ('y', '8', '9'), ('y', '?', 0), ('y', '3', '4'), ('B', '7', '8'), ('B', '?', 0), ('x', '11', '1'), ('x', '?', 0), ('i', '2', '3')}
RD[3]: {('y', '?', 0), ('y', '3', '4'), ('x', '0', '1'), ('B', '7', '8'), ('B', '?', 0), ('i', '?', 0), ('x', '11', '1'), ('y', '8', '9'), ('x', '?', 0), ('i', '2', '3')}
RD[4]: {('y', '?', 0), ('i', '10', '4'), ('y', '3', '4'), ('x', '0', '1'), ('B', '7', '8'), ('B', '?', 0), ('i', '?', 0), ('x', '11', '1'), ('x', '?', 0), ('i', '2', '3')}
RD[5]: {('y', '?', 0), ('i', '10', '4'), ('y', '3', '4'), ('x', '0', '1'), ('B', '7', '8'), ('B', '?', 0), ('i', '?', 0), ('x', '11', '1'), ('y', '8', '9'), ('x', '?', 0), ('i', '2', '3')}
RD[6]: {('i', '10', '4'), ('x', '0', '1'), ('i', '?', 0), ('y', '?', 0), ('y', '3', '4'), ('B', '7', '8'), ('B', '?', 0), ('x', '11', '1'), ('x', '?', 0), ('i', '2', '3')}
RD[7]: {('i', '10', '4'), ('x', '0', '1'), ('i', '?', 0), ('y', '?', 0), ('y', '3', '4'), ('B', '7', '8'), ('B', '?', 0), ('x', '11', '1'), ('x', '?', 0), ('i', '2', '3')}
RD[8]: {('i', '10', '4'), ('x', '0', '1'), ('i', '?', 0), ('y', '?', 0), ('y', '3', '4'), ('B', '7', '8'), ('B', '?', 0), ('x', '11', '1'), ('x', '?', 0), ('i', '2', '3')}
RD[9]: {('i', '10', '4'), ('x', '0', '1'), ('i', '?', 0), ('y', '8', '9'), ('y', '?', 0), ('B', '7', '8'), ('B', '?', 0), ('x', '11', '1'), ('x', '?', 0), ('i', '2', '3')}
RD[10]: {('i', '10', '4'), ('x', '0', '1'), ('i', '?', 0), ('y', '?', 0), ('y', '3', '4'), ('B', '7', '8'), ('B', '?', 0), ('x', '11', '1'), ('x', '?', 0), ('i', '2', '3')}
RD[11]: {('i', '10', '4'), ('x', '0', '1'), ('i', '?', 0), ('y', '8', '9'), ('y', '?', 0), ('y', '3', '4'), ('B', '7', '8'), ('B', '?', 0), ('x', '11', '1'), ('x', '?', 0), ('i', '2', '3')}
RD[12]: {('i', '10', '4'), ('x', '0', '1'), ('i', '?', 0), ('y', '8', '9'), ('y', '?', 0), ('y', '3', '4'), ('B', '7', '8'), ('B', '?', 0), ('x', '11', '1'), ('x', '?', 0), ('i', '2', '3')}
```

Exercise 2.18.

Proof by contradiction: assume $RD(q) \nsubseteq RD'(q)$ for some q.

Then there is a definition in RD which is not in RD'.

All the RD definitions come from this part of the algorithm:

while there exists an edge $(q_\circ, \alpha, q_\bullet) \in \mathbf{E}$
such that $(RD(q_\circ) \setminus \text{kill}_{RD}(q_\circ, \alpha, q_\bullet)) \cup \text{gen}_{RD}(q_\circ, \alpha, q_\bullet) \not\subseteq RD(q_\bullet)$
do $RD(q_\bullet) := RD(q_\bullet) \cup (RD(q_\circ) \setminus \text{kill}_{RD}(q_\circ, \alpha, q_\bullet)) \cup \text{gen}_{RD}(q_\circ, \alpha, q_\bullet)$

which essentially required for satisfying constraints, so if RD' does satisfies the constraints, then it must contains all elements of RD which is counter to our assumption.

Exercise 2.20:

For an arbitary array $A \in Arr$ and an arbitary path $\pi = q_0 a_1 q_1 a_2 q_3 \dots q_{n-1} a_n q_n$, we must prove that $Def(\pi, A) \subseteq RD(q_n)$.

From the definition of $Def(\pi, A)$, we got:

$$Def(\pi, A) = \{(A, ?, q_0)\} \cup \{(A, q_{i-1}, q_i) \mid A \in Def(\alpha_i)\}$$

α	$\text{kill}_{\text{RD}}(q_\circ, \alpha, q_\bullet)$	$\text{gen}_{\text{RD}}(q_\circ, \alpha, q_\bullet)$
$x := a$	$\{x\} \times \mathbf{Q}_? \times \mathbf{Q}$	$\{(x, q_\circ, q_\bullet)\}$
$A[a_1] := a_2$	$\{\}$	$\{(A, q_\circ, q_\bullet)\}$
$c?x$	$\{x\} \times \mathbf{Q}_? \times \mathbf{Q}$	$\{(x, q_\circ, q_\bullet)\}$
$c?A[a]$	$\{\}$	$\{(A, q_\circ, q_\bullet)\}$
$c!a$	$\{\}$	$\{\}$
b	$\{\}$	$\{\}$
skip	$\{\}$	$\{\}$

Since $(A, ?, q_D) \in RD(q_D]$ and $\forall j: (A, ?, q_D] \notin \text{kill}(a_j)$, so we can conclude that $(A, ?, q_D) \in RD(q_n)$.

$\forall j: A \in \text{Def}(a_j)$ it follows that $(A, q_{j-1}, q_j) \in \text{gen}_{RD}(q_{j-1}, a_j, q_j)$ and $(A, q_{j-1}, q_j) \notin \text{kill}(A, q_{j-1}, q_j)$, so we can conclude that $(A, q_{j-1}, q_j) \in RD(q_n]$.

So $\text{Def}(\pi, A) \subseteq RD(q_n]$

Q2:

LLVM (Low Level Virtual Machine) is a compiler infrastructure that provides a set of reusable libraries for building compilers, optimizers, and runtime environments. It offers a wide range of capabilities that make it suitable for various tasks in program analysis, optimization, and transformation. Here are some of the capabilities of LLVM:

1. Intermediate Representation (IR): LLVM uses an intermediate representation (IR) known as LLVM IR. This IR is designed to be language-agnostic and platform-independent. It serves as a common representation that can be used by different frontends (e.g., C, C++, Rust, Swift) to generate machine code.

2. Modular Design: LLVM is designed as a collection of reusable libraries that can be used independently or together. This modular design allows developers to use specific components of LLVM (e.g., the optimizer) in their own projects.

3. Optimizations: LLVM includes a suite of optimization passes that can improve the performance of generated code. These optimizations range from simple local transformations to more complex global optimizations.

4. Code Generation: LLVM supports code generation for various target architectures, including x86, ARM, MIPS, PowerPC, and more. It includes a backend infrastructure that can translate LLVM IR into machine code for the target architecture.

5. Just-In-Time (JIT) Compilation: LLVM can be used for JIT compilation, where code is compiled at runtime instead of ahead of time. This feature is particularly useful for dynamic languages and runtime optimization.

6. Debugging Support: LLVM includes support for generating debug information, which allows developers to debug optimized code using source-level debuggers.

7. Target-independent Features: LLVM provides several target-independent features, such as exception handling, garbage collection, and support for various calling conventions.

To encode a new language in LLVM, one typically follows these steps:

1. Design the Language: Define the syntax, semantics, and features of the new language. Decide how the language constructs map to LLVM IR.

2. Write a Frontend: Implement a frontend for the new language that translates source code written in the new language into LLVM IR. This frontend can be built using LLVM's libraries or using tools like LLVM's Kaleidoscope tutorial.

3. Define the Runtime: If the new language requires a runtime environment (e.g., for memory management, exception handling), define and implement the necessary runtime support using LLVM's facilities.

4. Optimize (Optional): Apply LLVM's optimization passes to the generated LLVM IR to improve the performance of the compiled code.

5. Generate Code: Use LLVM's code generation capabilities to translate the optimized LLVM IR into machine code for the target architecture.

Overall, LLVM provides a powerful infrastructure for building compilers and supporting various program analysis tasks, making it a popular choice for language designers and compiler developers.

Q3:

The program graph derivation process, as outlined in the appendices of the book "Program Analysis: An Appetizer," involves several steps:

1. Parsing: The program is parsed using a parser, which constructs an abstract syntax tree representing the program.

2. Abstract Syntax Tree: The abstract syntax tree represents the structure of the program in a hierarchical manner.

3. Constructing Program Graph: From the abstract syntax tree, a program graph is constructed. This process essentially involves implementing functions similar to those presented for the Guarded Commands language in Appendix A and developed for MicroC in the tasks of Appendix B. For generating control flow graphs (CFG) or data flow graphs (DFG), tools like LLVM can be utilized. However, for constructing the broader program graph, a specialized tool may be required.

4. Analysis: The constructed program graph is then subjected to analysis. This may involve tasks such as solving constraints and extracting constraints.

In summary, while LLVM can be useful for generating CFGs or DFGs, constructing the overall program graph typically requires a specialized tool, following the steps outlined in the book.

Q4:

RD_Analysis.py > ...

```python
import re
import networkx as nx

# Function to extract action name and (kill)/(gen) sets based on specific patterns
def extract_action_info(action_str):
    # Define regular expressions for each pattern
    patterns = {
        r'(\w+)\s*:=\s*(.+)$': 'assign',     # Pattern for x := a
        r'(\w+)\s*\[\s*(\w+)\s*\]\s*:=\s*(.+)$': 'array_assign',  # Pattern for A[a1] := a2
        r'(\w+)\s*\?\s*(\w+)': 'read',        # Pattern for c?x
        r'(\w+)\s*\?\s*(\w+)\s*\[\s*(\w+)\s*\]': 'array_read',    # Pattern for c?A[a]
        r'(\w+)\s*!\s*(\w+)': 'write'         # Pattern for c!a
    }



    # Iterate over patterns and check for a match
    for pattern, action_name in patterns.items():
        match = re.match(pattern, action_str)
        if match:
            if action_name == 'assign' or action_name == 'array_assign':
                return (action_name, match.groups()[0])
            if action_name == 'read' or action_name == 'array_read':
                return (action_name, match.groups()[1])
            if action_name == 'write':
                return (action_name, None)



    # List of operators and boolean values
    operators = ['=','!=', '<=', '>=', '<', '>', 'true', 'false', '!']

    # If no match found, check for operators and boolean values
    for op in operators:
        if op in action_str:
            return 'boolean', None  # Return 'boolean' with empty (kill) and (gen) sets


    # Treat 'skip' as a special action
    if action_str == 'skip':
        return 'skip', None  # Return 'skip' with empty (kill) and (gen) sets

    # If no match found, return None with empty (kill) and (gen) sets
    return None, None



# Function to build the directed graph from input file
def build_graph_from_file(file_name):
    G = nx.DiGraph()
    variables = set()
    arrays = set()
    with open(file_name, 'r') as file:
        num_nodes, num_edges = map(int, file.readline().split())
        edges = []
        for _ in range(num_edges):
            start_node, end_node, action_str = file.readline().split()
            edges.append((start_node, end_node))
            action_name, z = extract_action_info(action_str)
            G.add_edge(start_node, end_node, action=action_name, variable=z, kill={}, gen={})


    for start_node, end_node, data in G.edges(data=True):
        action_name = data['action']
        data['kill'] = set()  # Initialize kill set as empty set
        data['gen'] = set()   # Initialize gen set as empty set

        if action_name == 'assign':
            variables |= {data['variable']}
            data['kill'].update({(data['variable'], q0, q1) for q0, q1 in edges})
            data['kill'].add((data['variable'], -1, 0))
            data['gen'].update({(data['variable'], start_node, end_node)})
        elif action_name == 'array_assign':
            arrays |= {data['variable']}
            data['gen'].add((data['variable'], start_node, end_node))
        elif action_name == 'read':
            variables |= {data['variable']}
            data['kill'].update({(data['variable'], q0, q1) for q0, q1 in edges})
            data['kill'].add((data['variable'], -1, 0))
            data['gen'].update({(data['variable'], start_node, end_node)})
        elif action_name == 'array_read':
            arrays |= {data['variable']}
            data['gen'].add((data['variable'], start_node, end_node))
    G.graph['variables'] = variables
    G.graph['arrays'] = arrays
    return G, edges


def analysis(graph, edges):
    RD = [set() for _ in graph.nodes()]
    RD[0] |= {(x, '?', 0) for x in graph.graph['variables']}
    RD[0] |= {(x, '?', 0) for x in graph.graph['arrays']}



    while True:
```

```python
        flag = False
        for (q0, q1) in edges:
            if not (((RD[int(q0)] - graph.get_edge_data(q0, q1)['kill']) | graph.get_edge_data(q0, q1)['gen']) <= RD[int(q1)]):
                flag = True
                RD[int(q1)] = RD[int(q1)] | ((RD[int(q0)] - graph.get_edge_data(q0, q1)['kill']) | graph.get_edge_data(q0, q1)['gen'])
        if not flag:
            break



    for i, entry in enumerate(RD):
        print(f"RD[{i}]:", entry)


# Example usage
input_file = "pg1.txt"
graph, edges = build_graph_from_file(input_file)
analysis(graph, edges)
print('----------')

input_file = "pg2.txt"
graph, edges = build_graph_from_file(input_file)
analysis(graph, edges)

```

pg1.txt

```
5 5
0 1 y:=1
1 2 x>0
2 3 y:=x*y
3 1 x:=x-1
1 4 x<=0
```

pg2.txt

```
13 15
0 1 in?x
1 2 x>=0
2 3 i:=0
3 4 y:=-1
4 5 !(i<n)
5 11 out!y
11 1 in?x
10 4 i:=i+1
6 10 A[i]!=x
6 7 A[i]=x
7 8 B[i]:=B[i]+B[i]/10
8 9 y:=B[i]
9 5 skip
1 12 x<0
4 6 i<n
```

Results:

```
RD[0]: {('y', '?', 0), ('x', '?', 0)}
RD[1]: {('y', '2', '3'), ('y', '0', '1'), ('x', '3', '1'), ('y', '?', 0), ('x', '?', 0)}
RD[2]: {('x', '3', '1'), ('y', '2', '3'), ('y', '0', '1'), ('y', '?', 0), ('x', '?', 0)}
RD[3]: {('y', '2', '3'), ('x', '3', '1'), ('y', '?', 0), ('x', '?', 0)}
RD[4]: {('y', '2', '3'), ('y', '0', '1'), ('x', '3', '1'), ('y', '?', 0), ('x', '?', 0)}
----------
RD[0]: {('y', '?', 0), ('i', '?', 0), ('x', '?', 0), ('B', '?', 0)}
RD[1]: {('x', '0', '1'), ('i', '2', '3'), ('y', '3', '4'), ('B', '?', 0), ('B', '7', '8'), ('y', '8', '9'), ('x', '11', '1'), ('y', '?', 0), ('i', '?', 0), ('i', '10', '4'), ('x', '?', 0)}
RD[2]: {('x', '0', '1'), ('i', '2', '3'), ('B', '7', '8'), ('x', '11', '1'), ('i', '?', 0), ('i', '10', '4'), ('y', '8', '9'), ('y', '3', '4'), ('B', '?', 0), ('y', '?', 0), ('x', '?', 0)}
RD[3]: {('x', '0', '1'), ('i', '2', '3'), ('y', '3', '4'), ('B', '?', 0), ('B', '7', '8'), ('y', '8', '9'), ('x', '11', '1'), ('y', '?', 0), ('i', '?', 0), ('x', '?', 0)}
RD[4]: {('x', '0', '1'), ('i', '2', '3'), ('y', '3', '4'), ('B', '?', 0), ('B', '7', '8'), ('x', '11', '1'), ('y', '?', 0), ('i', '?', 0), ('i', '10', '4'), ('x', '?', 0)}
RD[5]: {('x', '0', '1'), ('i', '2', '3'), ('y', '3', '4'), ('B', '?', 0), ('B', '7', '8'), ('y', '8', '9'), ('x', '11', '1'), ('y', '?', 0), ('i', '?', 0), ('i', '10', '4'), ('x', '?', 0)}
RD[6]: {('x', '0', '1'), ('i', '2', '3'), ('B', '7', '8'), ('x', '11', '1'), ('i', '?', 0), ('i', '10', '4'), ('y', '3', '4'), ('B', '?', 0), ('y', '?', 0), ('x', '?', 0)}
RD[7]: {('x', '0', '1'), ('i', '2', '3'), ('B', '7', '8'), ('x', '11', '1'), ('i', '?', 0), ('i', '10', '4'), ('y', '3', '4'), ('B', '?', 0), ('y', '?', 0), ('x', '?', 0)}
RD[8]: {('x', '0', '1'), ('i', '2', '3'), ('B', '7', '8'), ('x', '11', '1'), ('i', '?', 0), ('i', '10', '4'), ('y', '3', '4'), ('B', '?', 0), ('y', '?', 0), ('x', '?', 0)}
RD[9]: {('x', '0', '1'), ('i', '2', '3'), ('B', '7', '8'), ('x', '11', '1'), ('i', '?', 0), ('i', '10', '4'), ('y', '8', '9'), ('B', '?', 0), ('y', '?', 0), ('x', '?', 0)}
RD[10]: {('x', '0', '1'), ('i', '2', '3'), ('B', '7', '8'), ('x', '11', '1'), ('i', '?', 0), ('i', '10', '4'), ('y', '3', '4'), ('B', '?', 0), ('y', '?', 0), ('x', '?', 0)}
RD[11]: {('x', '0', '1'), ('i', '2', '3'), ('B', '7', '8'), ('x', '11', '1'), ('i', '?', 0), ('i', '10', '4'), ('y', '8', '9'), ('y', '3', '4'), ('B', '?', 0), ('y', '?', 0), ('x', '?', 0)}
RD[12]: {('x', '0', '1'), ('i', '2', '3'), ('B', '7', '8'), ('x', '11', '1'), ('i', '?', 0), ('i', '10', '4'), ('y', '8', '9'), ('y', '3', '4'), ('B', '?', 0), ('y', '?', 0), ('x', '?', 0)}
```