Consider the following Datalog program $P$ (given in Logicblox syntax):

\[
\begin{align*}
oneway(x, y) & \rightarrow \text{int}(x), \text{int}(y). \\
path(x, y) & \rightarrow \text{int}(x), \text{int}(y). \\
edge(x, y) & \rightarrow \text{int}(x), \text{int}(y). \\
oneway(x, y) & \leftarrow \text{path}(x, y), \neg \text{path}(y, x). \\
path(x, y) & \leftarrow \text{edge}(x, y). \\
path(x, y) & \leftarrow \text{edge}(x, z), \text{path}(z, y).
\end{align*}
\]

In the program, the predicate $\text{edge}$ is input and the predicates $\text{oneway}$ and $\text{path}$ are derived. The set of possible constants that may appear in the predicates is fixed to $C = \{1, 2, 3\}$.

In this exercise, you will use the Z3 SMT solver to synthesize an input for the Datalog program above such that following queries hold:

\[
\begin{align*}
oneway(1, 2) \\
\neg \text{edge}(1, 2) \\
\neg \text{path}(2, 1)
\end{align*}
\]

**Solution** We observe that the Datalog program uses negation. We therefore first split the program into two strata:
Next, we encode each stratum using the SMT procedure given on slide 37. For unrolling the rules, we will use a bound of \( n = 2 \).

**Constraints for \( P_2 \)** Encoding the stratum \( P_2 \):

\[
\varphi_{P_2} \equiv (\forall X, Y. \text{oneway}_1(X, Y) \iff \text{path}(X, Y) \land \neg \text{path}(Y, X)) \\
\land (\forall X, Y. \text{oneway}_2(X, Y) \iff \text{path}(X, Y) \land \neg \text{path}(Y, X)) \\
\land (\forall X, Y. \text{oneway}(X, Y) \iff \text{path}(X, Y) \land \neg \text{path}(Y, X))
\]

Constraining the constants that may appear in the predicates of \( P_2 \):

\[
\varphi_{C_2} \equiv (\forall X, Y. (\text{oneway}_1(X, Y) \lor \text{oneway}_2(X, Y) \lor \text{oneway}(X, Y) \lor \text{path}(X, Y)) \\
\implies (X > 0 \land X < 4 \land Y > 0 \land Y < 4))
\]

Encoding the assertion for \( P_2 \):

\[
\varphi_{A_2} \equiv (\text{oneway}_2(1, 2) \land (\neg \text{path}(2, 1)))
\]

Here, we use \( \text{oneway}_2(1, 2) \) instead of \( \text{oneway}(1, 2) \) because this is a positive requirement and \( \text{oneway} \) is a derived predicate.

**Constraints for \( P_1 \)** Encoding the stratum \( P_1 \):

\[
\varphi_{P_1} \equiv (\forall X, Y. \text{path}_1(X, Y) \iff \text{edge}(X, Y)) \\
\land (\forall X, Y. \text{path}_2(X, Y) \iff \text{edge}(X, Y) \lor (\exists Z. \text{edge}(X, Z) \land \text{path}_1(Z, Y))) \\
\land (\forall X, Y. \text{path}(X, Y) \iff \text{edge}(X, Y)) \\
\land (\forall X, Y. \text{path}(X, Y) \iff \exists Z. \text{edge}(X, Z) \land \text{path}(Z, Y))
\]

Constraining the constants that may appear in the predicates of \( P_1 \):
\( \varphi_{C_1} \equiv (\forall X, Y. (\text{path}_1(X,Y) \lor \text{path}_2(X,Y) \lor \text{path}(X,Y) \lor \text{edge}(X,Y)) \land (X > 0 \land X < 4 \land Y > 0 \land Y < 4)) \)

Encoding the assertion for \( P_1 \):

\[ \varphi_{A_1} \equiv \neg \text{edge}(1,2) \land \neg \text{path}(2,1) \]

Here, we use \( \text{path}(2,1) \) because this is a negative requirement.

**Input synthesis**  We now iteratively synthesize an input for \( P \) by first synthesizing an input for \( P_2 \) (a set of path predicates) and then synthesizing an input for \( P_1 \) (a set of edge predicates) that is compatible with the input synthesized for \( P_2 \) (i.e., produces the set of path predicates synthesized for \( P_2 \)). We illustrate the steps taken to synthesize the input below.

**Step 1** Using Z3 (https://rise4fun.com/Z3) we find a model that satisfies the constraint

\[ \varphi_{P_2} \land \varphi_{C_2} \land \varphi_{A_2} \]

For the encoding of these above constraint in the SMT-LIB v2 format, see stratum2.txt. The synthesized input for \( P_2 \) is \( I^1_2 = \{\text{path}(1,2) \land \text{path}(1,3)\} \).

**Step 2** To synthesize input for \( P_1 \) that is compatible with the input \( I^1_2 \) synthesized in step 1, we encode the constraint:

\[ \varphi_{I^1_2} \equiv \forall X, Y. (((X = 1 \land Y = 2) \lor (X = 1 \land Y = 3)) \land \neg((X = 1 \land Y = 2) \lor (X = 1 \land Y = 3))) \Rightarrow \neg \text{path}(X,Y) \]

Here, for path predicates that must be derived we use \( \text{path}_2(X,Y) \) and for path predicates that must not be derived we use the predicate \( \text{path} \).

We try to find a model of the constraint:

\[ \varphi_{P_1} \land \varphi_{C_1} \land \varphi_{A_1} \land \varphi_{I^1_2} \]

This constraint is stored in stratum1_step2.txt. Unfortunately, such a model does not exist. We therefore backtrack to \( P_2 \) and try to synthesize an input other than \( I^1_2 \) that also satisfies the desired requirements for \( P_2 \).
Step 3. We encode an additional constraint that would ensure that we generate an input for $P_2$ that is different than $I_2^1$:

$$\varphi_{I_2^1} \equiv \neg \forall X, Y. (((X = 1 \land Y = 2) \lor (X = 1 \land Y = 3)) \implies path(X, Y))$$

$$\land (((\neg ((X = 1 \land Y = 2) \lor (X = 1 \land Y = 3))) \implies \neg path(X, Y))$$

Note that this constraint is similar to $\varphi_{I_2^1}$ in Step 2, except we had to negate the condition and to replace $path_2$ with $path$.

Next, we try to find a model of the following constraint:

$$\varphi_{P_2} \land \varphi_{C_2} \land \varphi_{A_2} \land \varphi_{I_2^1}$$

This constraint is stored in stratum2_step3.txt. This time, the synthesized input is $I_2^2 = \{path(1, 2) \land path(3, 1)\}$

Step 4. To synthesize input for $P_1$ that is compatible with the input $I_2^2$ synthesized in step 3, we encode the constraint:

$$\varphi_{I_2^2} \equiv \forall X, Y. (((X = 1 \land Y = 2) \lor (X = 3 \land Y = 1)) \implies path_2(X, Y))$$

$$\land (((\neg ((X = 1 \land Y = 2) \lor (X = 3 \land Y = 1)))) \implies \neg path(X, Y))$$

We try to find a model of the constraint:

$$\varphi_{P_1} \land \varphi_{C_1} \land \varphi_{A_1} \land \varphi_{I_2^1}$$

This constraint is stored in stratum1_step4.txt. This constraint is not satisfiable and again we have to backtrack to $P_2$ to synthesize an input different than $I_2^1$ and $I_2^2$ that satisfies the requirement.

Step 5. To ensure an input different than $I_2^3$ as well, we generate the constraint:

$$\varphi_{I_2^3} \equiv \forall X, Y. (((X = 1 \land Y = 2) \lor (X = 3 \land Y = 1)) \implies path(X, Y))$$

$$\land (((\neg ((X = 1 \land Y = 2) \lor (X = 3 \land Y = 1)))) \implies \neg path(X, Y))$$

We try to find a model of the constraint:

$$\varphi_{P_2} \land \varphi_{C_2} \land \varphi_{A_2} \land \varphi_{I_2^1} \land \varphi_{I_2^2}$$

The synthesized input is: $I_2^3 = \{path(1, 2) \land path(2, 3)\}$
**More steps**  \(I^3_2\) again cannot be produced by the stratum \(P_1\). The above steps are continued until we find an input for stratum \(P_2\) that can be output by \(P_1\). In our example, this happens, for instance, when the generated input for \(P_2\) is

\[I_2 = \{path(1, 3), path(3, 2), path(1, 2)\}\]

Then, the synthesized input to \(P_1\) is

\[I_1 = \{edge(1, 3), edge(3, 2)\}\]

The SMT constraints for the last two iterations of the above steps are given in the files `stratum2_final.txt` and `stratum1_final.txt`. 