Exercise 2

Input Synthesis for Datalog

Program Analysis for System Security and Reliability 2018
ETH Zurich

March 27, 2018

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

```
oneway(x, y) -> int(x), int(y).
path(x, y) -> int(x), int(y).
edge(x, y) -> int(x), int(y).

oneway(x,y) <- path(x,y), !path(y,x).
path(x, y) <- edge(x, y).
path(x, y) <- edge(x, z), path(z, y).
```

In the program, the predicate `edge` is input and the predicates `oneway` and `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:

```
oneway(1, 2)
!edge(1, 2)
!path(2, 1)
```

Solution We observe that the Datalog program uses negation. We therefore first split the program into two strata:

Stratum P_2	`oneway(x,y) <- path(x,y), !path(y,x)`
Stratum P_1	`path(x, y) <- edge(x, y)`
	`path(x, y) <- edge(x, z), path(z, y)`

Next, we encode each startum 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:

$$\begin{aligned}\varphi_{P_2} \equiv\ & \big(\forall X, Y.\ \mathit{oneway}_1(X,Y) \Leftrightarrow \mathit{path}(X,Y) \wedge \neg \mathit{path}(Y,X)\big) \\ & \wedge\ \big(\forall X, Y.\ \mathit{oneway}_2(X,Y) \Leftrightarrow \mathit{path}(X,Y) \wedge \neg \mathit{path}(Y,X)\big) \\ & \wedge\ \big(\forall X, Y.\ \mathit{oneway}(X,Y) \Leftarrow \mathit{path}(X,Y) \wedge \neg \mathit{path}(Y,X)\big)\end{aligned}$$

Constraining the constants that may appear in the predicates of P_2:

$$\begin{aligned}\varphi_{C_2} \equiv\ & \big(\forall X, Y.\ \big(\mathit{oneway}_1(X,Y) \vee \mathit{oneway}_2(X,Y) \vee \mathit{oneway}(X,Y) \vee \mathit{path}(X,Y)\big) \\ & \implies (X > 0 \wedge X < 4 \wedge Y > 0 \wedge Y < 4)\big)\end{aligned}$$

Encoding the assertion for P_2:

$$\varphi_{A_2} \equiv \big(\mathit{oneway}_2(1,2) \wedge (\neg \mathit{path}(2,1))\big)$$

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

Constraints for P_1 Encoding the stratum P_1:

$$\begin{aligned}\varphi_{P_1} \equiv\ & \big(\forall X, Y.\ \mathit{path}_1(X,Y) \Leftrightarrow \mathit{edge}(X,Y)\big) \\ & \wedge\big(\forall X, Y.\ \mathit{path}_2(X,Y) \Leftrightarrow \mathit{edge}(X,Y) \vee (\exists Z.\ \mathit{edge}(X,Z) \wedge \mathit{path}_1(Z,Y))\big) \\ & \wedge\big(\forall X, Y.\ \mathit{path}(X,Y) \Leftarrow \mathit{edge}(X,Y)\big) \\ & \wedge\big(\forall X, Y.\ \mathit{path}(X,Y) \Leftarrow \exists Z.\ \mathit{edge}(X,Z) \wedge \mathit{path}(Z,Y)\big)\end{aligned}$$

Constraining the constants that may appear in the predicates of P_1:

$$\begin{aligned}\varphi_{C_1} \quad \equiv \quad & \big(\forall X, Y.\ \big(path_1(X,Y) \vee path_2(X,Y) \vee path(X,Y) \vee edge(X,Y)\big) \\ & \implies (X > 0 \wedge X < 4 \wedge Y > 0 \wedge Y < 4)\big)\end{aligned}$$

Encoding the assertion for P_1:

$$\varphi_{A_1} \equiv \neg edge(1,2) \wedge \neg path(2,1)$$

Here, we use $path(2,1)$ 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

$$\varphi_{P_2} \wedge \varphi_{C_2} \wedge \varphi_{A_2}$$

For the encoding of these above constraint in the SMT-LIB v2 format, see `stratum2.txt`.

The synthesized input for P_2 is $I_2^1 = \{path(1,2) \wedge path(1,3)\}$.

Step 2 To synthesize input for P_1 that is compatible with the input I_2^1 synthesized in step 1, we encode the constraint:

$$\begin{aligned}\varphi_{I_2^1} \equiv \quad & \forall X, Y.\ (((X = 1 \wedge Y = 2) \vee (X = 1 \wedge Y = 3)) \implies path_2(X,Y)) \\ & \wedge((\neg((X = 1 \wedge Y = 2) \vee (X = 1 \wedge Y = 3))) \implies \neg path(X,Y))\end{aligned}$$

Here, for path predicates that must be derived we use $path_2(X,Y)$ and for path predicates that must not be derived we use the predicate *path*.

We try to find a model of the constraint:

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

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_2^1 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:

$$\begin{aligned}\varphi_{\neg I_2^1} \equiv \;& \neg\forall X, Y.\ (((X = 1 \wedge Y = 2) \vee (X = 1 \wedge Y = 3)) \implies path(X, Y)) \\ & \wedge((\neg((X = 1 \wedge Y = 2) \vee (X = 1 \wedge Y = 3))) \implies \neg path(X, Y))\end{aligned}$$

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:

$$\varphi_{P_2} \wedge \varphi_{C_2} \wedge \varphi_{A_2} \wedge \varphi_{\neg I_2^1}$$

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

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:

$$\begin{aligned}\varphi_{I_2^1} \equiv \;& \forall X, Y.\ (((X = 1 \wedge Y = 2) \vee (X = 3 \wedge Y = 1)) \implies path_2(X, Y)) \\ & \wedge((\neg((X = 1 \wedge Y = 2) \vee (X = 3 \wedge Y = 1))) \implies \neg path(X, Y))\end{aligned}$$

We try to find a model of the constraint:

$$\varphi_{P_1} \wedge \varphi_{C_1} \wedge \varphi_{A_1} \wedge \varphi_{I_2^2}$$

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_2^2 as well, we generate the constraint:

$$\begin{aligned}\varphi_{\neg I_2^2} \equiv \;& \neg\forall X, Y.\ (((X = 1 \wedge Y = 2) \vee (X = 3 \wedge Y = 1)) \implies path(X, Y)) \\ & \wedge((\neg((X = 1 \wedge Y = 2) \vee (X = 3 \wedge Y = 1))) \implies \neg path(X, Y))\end{aligned}$$

We try to find a model of the constraint:

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

The synthesized input is: $I_2^3 = \{path(1, 2) \wedge path(2, 3)\}$

More steps I_2^3 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`.