# Supplementary Material

## 1 Q1

In this section, we illustrate all the logic bugs detected by DEOPT.

### 1.1 Bug 1

Bug 1 was found in Soufflé. In interpreter mode of Soufflé, `0.0` not equals to `-0.0`, because they use bitwise comparison for float type. This bug was found due to an inconsistency, as the presence of **`inline`** keyword changed the result from `-0` to `0`. And we checked that when we remove the **`inline`** keyword or replaced it with **`magic`** keyword, the final result equaled to `-0`, which means the **`inline`** keyword result in this inconsistent behavior. Here, we omit the **`inline`** keyword for the reference program, as Souffle does not support inline optimization for input and output relations. However, we think that our test oracle can block the inline optimization if we can add **`inline`** keyword for reference programs.

**Table 1: Bug 1 found by DEOPT.**

| iteration | $P_n^{ref}$ | $Facts_{output\_rel}^{ref}$ | $P_n^{opt}$ | $Facts_{output\_rel}^{opt}$ |
|---|---|---|---|---|
| 0 | `.decl olkw(A:float)`<br>`.decl kkcb(A:float)`<br>`.output kkcb`<br>`olkw(-0).`<br>`kkcb(A) :- olkw(A).` | `kkcb(-0).` | `.decl olkw(A:float)`<br>`.decl kkcb(A:float)`<br>`.output kkcb`<br>`olkw(-0).`<br>`kkcb(A) :- olkw(A).` | `kkcb(-0).` |
| 1 | `.decl kkcb(A:float)`<br>`.decl nvuk(A:float)`<br>`.output nvuk`<br>`kkcb(-0).`<br>`nvuk(A) :- kkcb(A), 0.0 = A.` | `nvuk(-0).` | `.decl olkw(A:float)`<br>`.decl kkcb(A:float) inline`<br>`.decl nvuk(A:float)`<br>`.output nvuk`<br>`olkw(-0).`<br>`kkcb(A) :- olkw(A).`<br>`nvuk(A) :- kkcb(A), 0.0 = A.` | `nvuk(0).` |

### 1.2 Bug 2

Bug 2 is much similar with bug 1. In this test case, the inconsistency was only caused by **`magic`** transformation. Based on this, we determined they were different unique bugs, and also different from bug 5 found by queryFuzz shown in Section 2.1.5. To demonstrate the effectiveness of our method, we did not remove the **`magic`** keyword in the reference program.

**Table 2: Bug 2 found by DEOPT.**

| iteration | $P_n^{ref}$ | $Facts_{output\_rel}^{ref}$ | $P_n^{opt}$ | $Facts_{output\_rel}^{opt}$ |
|---|---|---|---|---|
| 0 | `.decl yona(A:symbol, C:float)`<br>`.decl idck(B:float) magic`<br>`.output idck`<br>`yona("DrdskfIQ9A", 0).`<br>`idck(A) :- yona(D, A).` | `idck(0).` | `.decl yona(A:symbol, C:float)`<br>`.decl idck(B:float) magic`<br>`.output idck`<br>`yona("DrdskfIQ9A", 0).`<br>`idck(A) :- yona(D, A).` | `idck(0).` |
| 1 | `.decl idck(B:float) magic`<br>`.decl zsjo(A:float)`<br>`.output zsjo`<br>`idck(0).`<br>`zsjo(D) :- idck(D), idck(min(-D,D)).` | `zsjo(0).` | `.decl yona(A:symbol, C:float)`<br>`.decl idck(B:float) magic`<br>`.decl zsjo(A:float)`<br>`.output zsjo`<br>`yona("DrdskfIQ9A", 0).`<br>`idck(A) :- yona(D, A).`<br>`zsjo(D) :- idck(D), idck(min(-D,D)).` | - |

## 1.3 Bug 3

Bug 3 was a bug in Soufflé, related to subsumption and `magic` transformation. For this bug, we thought that queryFuzz was difficult to detect. queryFuzz tests queries which beyond conjunctive queries based on the monotonicity of conjunctive queries. Subsumption is used to remove some facts of a relation, based on the conditions in its body. If no facts meet these conditions, the number of facts will not change. So it is difficult to predict how the number of facts will change after subsumption. Adding or removing facts to or from the relation in subsumption may leave the results of this relation unchanged after the subsumption. Because we cannot tell whether the added or deleted facts are the ones to be removed by the subsumption. Such as we want to add facts to the EDB, we can only assume that the old results are a subset of the new results, and cannot tell if they should be equal. But in this bug, subsumption not works under `magic` transformation, which means it will remove nothing from the relation. This does not contradict the old result is the subset of the new result. So queryFuzz can not detect this bug.

**Table 3: Bug 3 found by DEOPT.**

| iteration | $P_n^{ref}$ | $Facts_{output\_rel}^{ref}$ | $P_n^{opt}$ | $Facts_{output\_rel}^{opt}$ |
|---|---|---|---|---|
| 0 | `.decl buyh(A:unsigned) magic`<br>`.decl eozn(A:unsigned)`<br>`.output eozn`<br>`buyh(6). buyh(7). buyh(3).`<br>`eozn(E) :- buyh(E).` | `eozn(6).`<br>`eozn(7).`<br>`eozn(3).` | `.decl buyh(A:unsigned) magic`<br>`.decl eozn(A:unsigned)`<br>`.output eozn`<br>`buyh(6). buyh(7). buyh(3).`<br>`eozn(E) :- buyh(E).` | `eozn(6).`<br>`eozn(7).`<br>`eozn(3).` |
| 1 | `.decl eozn(A:unsigned)`<br>`.output eozn`<br>`eozn(6). eozn(7). eozn(3).`<br>`eozn(E1) <= eozn(E2) :- E1<E2.` | `eozn(7).` | `.decl buyh(A:unsigned) magic`<br>`.decl eozn(A:unsigned)`<br>`.output eozn`<br>`buyh(6). buyh(7). buyh(3).`<br>`eozn(E) :- buyh(E).`<br>`eozn(E1) <= eozn(E2) :- E1<E2.` | `eozn(6).`<br>`eozn(7).`<br>`eozn(3).` |

## 1.4 Bug 4

Bug 4 was a bug in Soufflé, related to subsumption and `magic` transformation. The difference from bug 3 was that in this test case, the results of subsumption were correct. Similar with bug 3, queryFuzz was also unable to detect this bug due to the same underlying reason.

**Table 4: Bug 4 found by DEOPT.**

| iteration | $P_n^{ref}$ | $Facts_{output\_rel}^{ref}$ | $P_n^{opt}$ | $Facts_{output\_rel}^{opt}$ |
|---|---|---|---|---|
| 0 | `.decl kdof(A:unsigned, B:number)`<br>`.output kdof`<br>`kdof(1, -2). kdof(1, 9).`<br>`kdof(B, A1) <= kdof(B, A2) :- A1<A2.` | `kdof(1, 9).` | `.decl kdof(A:unsigned, B:number)`<br>`.output kdof`<br>`kdof(1, -2). kdof(1, 9).`<br>`kdof(B, A1) <= kdof(B, A2) :- A1<A2.` | `kdof(1, 9).` |
| 1 | `.decl kdof(A:unsigned, B:number)`<br>`.decl ovqb(A:number, B:unsigned) magic`<br>`.output khkw`<br>`kdof(1, 9).`<br>`ovqb(A^-3, B) :- kdof(B, A).` | `ovqb(0, 1).` | `.decl kdof(A:unsigned, B:number)`<br>`.decl ovqb(A:number, B:unsigned) magic`<br>`.decl khkw(A:number, B:unsigned)`<br>`.output ovqb`<br>`kdof(1, -2).`<br>`kdof(1, 9).`<br>`kdof(B, A1) <= kdof(B, A2) :- A1<A2.`<br>`ovqb(A^-3, B) :- kdof(B, A).` | `ovqb(0, 1).` |
| 2 | `.decl kdof(A:unsigned, B:number)`<br>`.decl ovqb(A:number, B:unsigned) magic`<br>`.decl khkw(A:number, B:unsigned)`<br>`.output khkw`<br>`kdof(1, 9). ovqb(0, 1).`<br>`khkw(A, B) :- kdof(B, A), !ovqb(A, B).` | `khkw(9, 1).` | `.decl kdof(A:unsigned, B:number)`<br>`.decl ovqb(A:number, B:unsigned) magic`<br>`.decl khkw(A:number, B:unsigned)`<br>`.output khkw`<br>`kdof(1, -2). kdof(1, 9).`<br>`kdof(B, A1) <= kdof(B, A2) :- A1<A2.`<br>`ovqb(A^-3, B) :- kdof(B, A).`<br>`khkw(A, B) :- kdof(B, A), !ovqb(A, B).` | `khkw(-2, 1).`<br>`khkw(9, 1).` |

## 1.5 Bug 5

Bug 5 was a bug in Soufflé, related to equivalence relations and `magic` transformation. Equivalence relation is a binary relation in Soufflé and exhibits three properties: reflexivity, symmetry, and transitivity. So when given an input to the equivalence relation, that input should be a subset of the final result, but there is no way to tell if they are equal. This bug caused by that equivalence relation not works under `magic` transformation. So similar with bug 3, queryFuzz can not detect this bug. This bug also could be triggered with `inline` transformation.

**Table 5: Bug 5 found by DEOPT.**

| iteration | $P_n^{ref}$ | $Facts_{output\_rel}^{ref}$ | $P_n^{opt}$ | $Facts_{output\_rel}^{opt}$ |
|---|---|---|---|---|
| 0 | `.decl a(c:unsigned)`<br>`.decl d(b:unsigned, c:unsigned) magic eqrel`<br>`.output d`<br>`a(8). a(4).`<br>`d(f+4, f) :- a(f).` | `d(4, 4).`<br>`d(4, 8).`<br>`d(4, 12).`<br>`d(8, 4).`<br>`d(8, 8).`<br>`d(8, 12).`<br>`d(12, 4).`<br>`d(12, 8).`<br>`d(12, 12).` | `.decl a(c:unsigned)`<br>`.decl d(b:unsigned, c:unsigned) magic eqrel`<br>`.output d`<br>`a(8). a(4).`<br>`d(f+4, f) :- a(f).` | `d(4, 4).`<br>`d(4, 8).`<br>`d(4, 12).`<br>`d(8, 4).`<br>`d(8, 8).`<br>`d(8, 12).`<br>`d(12, 4).`<br>`d(12, 8).`<br>`d(12, 12).` |
| 1 | `.decl a(c:unsigned)`<br>`.decl d(b:unsigned, c:unsigned) magic eqrel`<br>`.decl bxax(e:unsigned)`<br>`.output bxax`<br>`a(8). a(4). d(4, 4). d(4, 8). d(4, 12).`<br>`d(8, 4). d(8, 8). d(8, 12). d(12, 4).`<br>`d(12, 8). d(12, 12).`<br>`bxax(c) :- d(c, e), a(c).` | `bxax(4).`<br>`bxax(8).` | `.decl a(c:unsigned)`<br>`.decl d(b:unsigned, c:unsigned) magic eqrel`<br>`.decl bxax(e:unsigned)`<br>`.output bxax`<br>`a(8). a(4).`<br>`d(f+4, f) :- a(f).`<br>`bxax(c) :- d(c, e), a(c).` | `bxax(8).` |

## 1.6 Bug 6

Bug 6 was a bug in Soufflé, related to the conflict of `inline` and `no_magic` keywords.

**Table 6: Bug 6 found by DEOPT.**

| iteration | $P_n^{ref}$ | $Facts_{output\_rel}^{ref}$ | $P_n^{opt}$ | $Facts_{output\_rel}^{opt}$ |
|---|---|---|---|---|
| 0 | `.decl a(A:number)`<br>`.decl b(A:number)`<br>`.output b`<br>`a(-1). a(1). a(2).`<br>`b(A) :- a(A), a(-A).` | `b(1).`<br>`b(-1).` | `.decl a(A:number)`<br>`.decl b(A:number)`<br>`.output b`<br>`a(-1). a(1). a(2).`<br>`b(A) :- a(A), a(-A).` | `b(1).`<br>`b(-1).` |
| 1 | `.decl b(A:number)`<br>`.decl c(A:number) no_magic`<br>`.output c`<br>`b(1). b(-1).`<br>`c(A) :- b(A).` | `c(1).`<br>`c(-1).` | `.decl a(A:number)`<br>`.decl b(A:number) inline`<br>`.decl c(A:number) no_magic`<br>`.output c`<br>`a(-1). a(1). a(2).`<br>`b(A) :- a(A), a(-A).`<br>`c(A) :- b(A).` | `c(-1).`<br>`c(1).`<br>`c(2).` |

## 1.7 Bug 7

Bug 7 was a bug in Soufflé, it produced different results under interpreter mode and synthesizer mode. In interpreter mode, it could produce the correct results, but in synthesizer mode, it produced empty results. Only when execute the test case with `-c` option, we can trigger this bug.

```
.decl payb(A:float, B:float)
.decl uuyd(A:float)
.output uuyd

payb(0.80300000000000082, 0.80300000000000082).
uuyd(F) :- payb(0.80300000000000082, F).

// expected result:
// uuyd(0.80300000000000082).
// buggy result:
// -
```

**Figure 1: Bug 7 found by DEOPT.**

## 1.8 Bug 8

Bug 8 was a bug in Soufflé, equivalence relation not worked when we applied provenance to the program, which was controlled by execution option `-t`. Provenance is utilized to furnish insight for troubleshooting Datalog programs. This bug was triggered under two specific execution options: `-t none` and `--disable-transformers=ExpandEqrelsTransformer`. Similar with bug 3, queryFuzz can not detect this bug.

```
.decl ytfh(A:float)
.decl bkbi(A:float, B:float) eqrel
.output bkbi

ytfh(1).

bkbi(E, E+1) :- ytfh(E).
// expected result:
// bkbi(1, 2). bkbi(1, 1). bkbi(2, 2). bkbi(2, 1).
// buggy result:
// bkbi(1, 2).
```

**Figure 2: Bug 8 found by DEOPT.**

## 1.9 Bug 9

Bug 9 was a bug in Soufflé, related to `subsumption`.

**Table 7: Bug 9 found by DEOPT.**

| iteration | $P_n^{ref}$ | $Facts_{output\_rel}^{ref}$ | $P_n^{opt}$ | $Facts_{output\_rel}^{opt}$ |
|---|---|---|---|---|
| 0 | `.decl a(A:number)`<br>`.output a`<br>`a(1). a(2).`<br>`a(A1)<=a(A2):-A1<A2.` | `a(2).` | `.decl a(A:number)`<br>`.output a`<br>`a(1). a(2).`<br>`a(A1)<=a(A2):-A1<A2.` | `a(2).` |
| 1 | `.decl a(A:number)`<br>`.output a`<br>`a(2).`<br>`a(A1)<=a(A2):-A1>A2.` | `a(2).` | `.decl a(A:number)`<br>`.output a`<br>`a(1). a(2).`<br>`a(A1)<=a(A2):-A1<A2.`<br>`a(A1)<=a(A2):-A1>A2.` | - |

## 1.10 Bug 10

Bug 10 was a bug in μZ. The developer fixed this bug after we report it, but did not give us any feedback about this bug. This was an interesting bug that all the four rules in this test case did not depend on each other, but each of the first three rules could have an effect on the results of this bug-inducing rule. Also, both of the results of test oracle and test case were wrong.

**Table 8: Bug 10 found by DEOPT.**

| iteration | $P_n^{ref}$ | $Facts_{output\_rel}^{ref}$ | $P_n^{opt}$ | $Facts_{output\_rel}^{opt}$ |
|---|---|---|---|---|
| 0 | `Z 128`<br>`mxsr(A:Z) input`<br>`ebbj(A:Z) printtuples`<br>`mxsr(1).`<br>`ebbj(A) :- mxsr(A), 43 != A.` | `ebbj(1).` | `Z 128`<br>`mxsr(A:Z) input`<br>`qjfp(A:Z) input`<br>`jrkr(A:Z, B:Z) input`<br>`rtkv(A:Z) input`<br>`ebbj(A:Z) printtuples`<br>`jrkr(80, 80). jrkr(29, 29).`<br>`jrkr(4, 4). mxsr(1).`<br>`qjfp(1). rtkv(1).`<br>`ebbj(A) :- mxsr(A), 43 != A.` | `ebbj(1).` |
| 1 | `Z 128`<br>`qjfp(A:Z) input`<br>`oxyx(A:Z) printtuples`<br>`qjfp(1).`<br>`oxyx(C) :- qjfp(C), 76 != C.` | `oxyx(1).` | `Z 128`<br>`mxsr(A:Z) input`<br>`qjfp(A:Z) input`<br>`jrkr(A:Z, B:Z) input`<br>`rtkv(A:Z) input`<br>`ebbj(A:Z)`<br>`oxyx(A:Z) printtuples`<br>`jrkr(80, 80). jrkr(29, 29).`<br>`jrkr(4, 4). mxsr(1).`<br>`qjfp(1). rtkv(1).`<br>`ebbj(A) :- mxsr(A), 43 != A.`<br>`oxyx(C) :- qjfp(C), 76 != C.` | `oxyx(1).` |
| 2 | `Z 128`<br>`rtkv(A:Z) input`<br>`iypi(A:Z) printtuples`<br>`rtkv(1).`<br>`iypi(A) :- rtkv(A), 77 < A.` | `iypi(1).` | `Z 128`<br>`mxsr(A:Z) input`<br>`qjfp(A:Z) input`<br>`jrkr(A:Z, B:Z) input`<br>`rtkv(A:Z) input`<br>`ebbj(A:Z)`<br>`oxyx(A:Z)`<br>`iypi(A:Z) printtuples`<br>`jrkr(80, 80). jrkr(29, 29).`<br>`jrkr(4, 4). mxsr(1).`<br>`qjfp(1). rtkv(1).`<br>`ebbj(A) :- mxsr(A), 43 != A.`<br>`oxyx(C) :- qjfp(C), 76 != C.`<br>`iypi(A) :- rtkv(A), 77 < A.` | `iypi(1).` |
| 3 | `Z 128`<br>`jrkr(A:Z, B:Z) input`<br>`fvof(A:Z) printtuples`<br>`jrkr(80, 80).`<br>`jrkr(29, 29).`<br>`jrkr(4, 4).`<br>`fvof(E) :- jrkr(D, E), 8 != E, 71 < D.` | `fvof(29).`<br>`fvof(80).`<br>`fvof(4).` | `Z 128`<br>`mxsr(A:Z) input`<br>`qjfp(A:Z) input`<br>`jrkr(A:Z, B:Z) input`<br>`rtkv(A:Z) input`<br>`ebbj(A:Z)`<br>`oxyx(A:Z)`<br>`iypi(A:Z)`<br>`fvof(A:Z) printtuples`<br>`jrkr(80, 80). jrkr(29, 29).`<br>`jrkr(4, 4). mxsr(1).`<br>`qjfp(1). rtkv(1).`<br>`ebbj(A) :- mxsr(A), 43 != A.`<br>`oxyx(C) :- qjfp(C), 76 != C.`<br>`iypi(A) :- rtkv(A), 77 < A.`<br>`fvof(E) :- jrkr(D, E), 8 != E, 71 < D.` | `fvof(29).`<br>`fvof(80).` |

## 1.11 Bug 11

Bug 11 was a bug in μZ. Similar with the bug 10, the rules in this test case did not depend on each other. The reference program in this example trigger the bug.

**Table 9: Bug 11 found by DEOPT.**

| iteration | $P_n^{ref}$ | $Facts_{output\_rel}^{ref}$ | $P_n^{opt}$ | $Facts_{output\_rel}^{opt}$ |
|---|---|---|---|---|
| 0 | `Z 128`<br>`hmyo(A:Z) input`<br>`cdan(A:Z) printtuples`<br>`hmyo(70).`<br>`cdan(C) :- hmyo(C), 65 < C.` | `cdan(70).` | `Z 128`<br>`szra(A:Z) input`<br>`hmyo(A:Z) input`<br>`cdan(A:Z) printtuples`<br>`szra(1).`<br>`hmyo(70).`<br>`cdan(C) :- hmyo(C), 65 < C.` | `cdan(70).` |
| 1 | `Z 128`<br>`szra(A:Z) input`<br>`fbnd(A:Z) printtuples`<br>`szra(1).`<br>`fbnd(F) :- szra(F), 72 != F, 97 = F.` | `fbnd(1).` | `Z 128`<br>`szra(A:Z) input`<br>`hmyo(A:Z) input`<br>`cdan(A:Z)`<br>`fbnd(A:Z) printtuples`<br>`szra(1).`<br>`hmyo(70).`<br>`cdan(C) :- hmyo(C), 65 < C.`<br>`fbnd(F) :- szra(F), 72 != F, 97 = F.` | - |

## 1.12 Bug 12

Bug 12 was a bug in CozoDB, which was caused by an error in the implementation of semi-naïve algorithm.

**Table 10: Bug 12 found by DEOPT.**

| iteration | $P_n^{ref}$ | $Facts_{output\_rel}^{ref}$ | $P_n^{opt}$ | $Facts_{output\_rel}^{opt}$ |
|---|---|---|---|---|
| 0 | `x[A] := A = 1`<br>`?[A] := x[A]` | `[[1]]` | `x[A] := A = 1`<br>`?[A] := x[A]` | `[[1]]` |
| 1 | `y[A, A] := A = 1`<br>`?[A, B] := y[A, B]` | `[[1, 1]]` | `x[A] := A = 1`<br>`y[A, A] := A = 1`<br>`?[A, B] := y[A, B]` | `[[1, 1]]` |
| 2 | `x[A] <- [[1]]`<br>`y[A, B] := A = 0, B = 1, x[B]`<br>`?[A, B] := y[A, B]` | `[[0, 1]]` | `x[A] := A = 1`<br>`y[A, A] := A = 1`<br>`y[A, B] := A = 0, B = 1, x[B]`<br>`?[A, B] := y[A, B]` | `[[0, 1]]` |
| 3 | `y[A, B] <- [[0, 1], [1, 1]]`<br>`?[C] := y[A, _], y[C, A]` | `[[0], [1]]` | `x[A] := A = 1`<br>`y[A, A] := A = 1`<br>`y[A, B] := A = 0, B = 1, x[B]`<br>`?[C] := y[A, _], y[C, A]` | `[[1]]` |

## 1.13 Bug 13

Bug 13 was a bug in CozoDB, which was caused by the discrepancy between the comparison functions employed in the magic sets rewrite and the binary operators utilized in the rules. We anticipated the results to be `[[0.0, null]]`, but CozoDB returned `[[0.0, null], [-0.0, null]]`, where the second result violates the constraint `A >= 0`. During the magic sets rewrite, CozoDB employs range scanning on the relation `a` instead of testing each individual value. However, the comparison function used in the range scan considers `-0.0 >= 0` to be true, while the comparison function used in the binary operators considers it to be false. This bug can only be identified when the program is evaluated twice, once with the magic sets rewrite and once without it. We believe that queryFuzz would not be able to detect this bug because the transformations it applies are designed specifically for conjunctive queries, which consist of single non-recursive function-free Horn rules. We got feedback from the developer of CozoDB that their magic sets rewrite only applied on conjunctive queries. Therefore, the transformations supported by queryFuzz, such as adding an existing subgoal into the rule body, modifying a variable, or removing a subgoal from the rule body, are inadequate for preventing the magic sets rewrite in this particular scenario.

**Table 11: Bug 13 found by DEOPT.**

| iteration | $P_n^{ref}$ | $Facts_{output\_rel}^{ref}$ | $P_n^{opt}$ | $Facts_{output\_rel}^{opt}$ |
|---|---|---|---|---|
| 0 | `phve[A, B] <- [[-0.0, null], [0.0, null]]`<br>`xukw[A, B] := phve[A, B], ge(A, 0)`<br>`?[A, B] := xukw[A, B]` | `[[0.0, null]]` | `phve[A, B] <- [[-0.0, null], [0.0, null]]`<br>`xukw[A, B] := phve[A, B], ge(A, 0)`<br>`?[A, B] := xukw[A, B]` | `[[0.0, null]]` |
| 1 | `phve[A, B] <- [[-0.0, null], [0.0, null]]`<br>`nwku[F] := phve[_, F]`<br>`?[A] := nwku[A]` | `[[null]]` | `phve[A, B] <- [[-0.0, null], [0.0, null]]`<br>`xukw[A, B] := phve[A, B], ge(A, 0)`<br>`nwku[F] := phve[_, F]`<br>`?[A] := nwku[A]` | `[[null]]` |
| 2 | `xukw[A, B] <- [[0.0, null]]`<br>`nwku[A] <- [[null]]`<br>`ssie[F, C] := nwku[C], xukw[F, C]`<br>`?[A, B] := ssie[A, B]` | `[[0.0, null]]` | `phve[A, B] <- [[-0.0, null], [0.0, null]]`<br>`xukw[A, B] := phve[A, B], ge(A, 0)`<br>`nwku[F] := phve[_, F]`<br>`ssie[F, C] := nwku[C], xukw[F, C]`<br>`?[A, B] := ssie[A, B]` | `[[0.0, null],`<br>`[-0.0, null]]` |

## 2 Q2

In this section, we demonstrate the best-effort comparison based on manually inspecting and analyzing the bug-inducing test cases and the bugs. We first checkout to the version of Datalog engine which contains the bug to confirm that the bug can be reproduced. Then we emulated our process of generating test oracles on all the logic bugs found by queryFuzz. For the bugs found by Deopt, we analyze the reasons why it might not be detected by queryFuzz.

### 2.1 Bugs found by queryFuzz

*2.1.1 Bug 1.* Through the description in the GitHub issue of this bug we found that the bug 1 was related to magic transformation, which was used to minimize the program. But queryFuzz provided too much inputs, we picked the minimum inputs that triggered the bug. We can see that in the $iteration_4$, the result of $P_4^{opt}$ was different from that of $P_4^{ref}$.

**Table 12: Bug 1 found by queryFuzz.**

| iteration | $P_n^{ref}$ | $Facts_{output\_rel}^{ref}$ | $P_n^{opt}$ | $Facts_{output\_rel}^{opt}$ |
|---|---|---|---|---|
| 0 | `.decl MZV(U:number, V:number)`<br>`.decl HqV(U:number)`<br>`.output HqV`<br>`MZV(9, 28). MZV(18, 18).`<br>`HqV(a) :- MZV(a,b).` | `HqV(9).`<br>`HqV(18).` | `.decl MZV(U:number, V:number)`<br>`.decl HqV(U:number)`<br>`.output HqV`<br>`MZV(9, 28). MZV(18, 18).`<br>`HqV(a) :- MZV(a,b).` | `HqV(9).`<br>`HqV(18).` |
| 1 | `.decl MZV(U:number, V:number)`<br>`.decl gQk(U:number)`<br>`.output gQk`<br>`MZV(9, 28). MZV(18, 18).`<br>`gQk(jW) :- MZV(jW,jW).` | `gQk(18).` | `.decl MZV(U:number, V:number)`<br>`.decl HqV(U:number)`<br>`.decl gQk(U:number)`<br>`.output gQk`<br>`MZV(9, 28). MZV(18, 18).`<br>`HqV(a) :- MZV(a,b).`<br>`gQk(jW) :- MZV(jW,jW).` | `gQk(18).` |
| 2 | `.decl MZV(U:number, V:number)`<br>`.decl HqV(U:number)`<br>`.decl gQk(U:number)`<br>`.decl QOq(U:number, V:number)`<br>`MZV(9, 28). MZV(18, 18).`<br>`HqV(9). HqV(18). gQk(18).`<br>`QOq(aS,GF) :- MZV(GF,GF), gQk(M),`<br>`HqV(aS), MZV(aS,M).`<br>`.output QOq` | `QOq(18, 18).` | `.decl MZV(U:number, V:number)`<br>`.decl HqV(U:number)`<br>`.decl gQk(U:number)`<br>`.decl QOq(U:number, V:number)`<br>`.output QOq`<br>`MZV(9, 28). MZV(18, 18).`<br>`HqV(a) :- MZV(a,b).`<br>`gQk(jW) :- MZV(jW,jW).`<br>`QOq(aS,GF) :- MZV(GF,GF), gQk(M),`<br>`HqV(aS), MZV(aS,M).` | `QOq(18, 18).` |
| 3 | `.decl gQk(U:number)`<br>`.decl QOq(U:number, V:number)`<br>`.decl RwL(U:number)`<br>`.output RwL`<br>`gQk(18). QOq(18, 18).`<br>`RwL(qr) :- QOq(u,qr), gQk(u), gQk(u).` | `RwL(18).` | `.decl MZV(U:number, V:number)`<br>`.decl HqV(U:number)`<br>`.decl gQk(U:number)`<br>`.decl QOq(U:number, V:number)`<br>`.decl RwL(U:number)`<br>`.output RwL`<br>`MZV(9, 28). MZV(18, 18).`<br>`HqV(a) :- MZV(a,b).`<br>`gQk(jW) :- MZV(jW,jW).`<br>`QOq(aS,GF) :- MZV(GF,GF), gQk(M),`<br>`HqV(aS), MZV(aS,M).`<br>`RwL(qr) :- QOq(u,qr), gQk(u), gQk(u).` | `RwL(18).` |
| 4 | `.decl MZV(U:number, V:number)`<br>`.decl gQk(U:number)`<br>`.decl RwL(U:number)`<br>`.decl out(U:number, V:number)`<br>`.output out`<br>`MZV(9, 28). MZV(18, 18).`<br>`gQk(18). RwL(18).`<br>`out(jB,ym) :- gQk(h), RwL(ym), MZV(h,jB).` | `out(18, 18).` | `.decl MZV(U:number, V:number)`<br>`.decl HqV(U:number)`<br>`.decl gQk(U:number)`<br>`.decl QOq(U:number, V:number)`<br>`.decl RwL(U:number)`<br>`.decl out(U:number, V:number)`<br>`.output out`<br>`MZV(9, 28). MZV(18, 18).`<br>`HqV(a) :- MZV(a,b).`<br>`gQk(jW) :- MZV(jW,jW).`<br>`QOq(aS,GF) :- MZV(GF,GF), gQk(M),`<br>`HqV(aS), MZV(aS,M).`<br>`RwL(qr) :- QOq(u,qr), gQk(u), gQk(u).`<br>`out(jB,ym) :- gQk(h), RwL(ym), MZV(h,jB).` | `out(18, 18).`<br>`out(28, 18).` |

*2.1.2 Bug 2.* The bug 2 was caused by magic transformation, in the same time, the inputs of Datalog program must come from an input file. So we needed two options, which were the necessary conditions for triggering the bug. The first one was `-F`, which means to specify the path where the input file is located, we also created an input file named `IBf.facts` which only contains one entry `1`. The second was `--magic-transform=*`, which means to perform magic transformation on all the relations in the program. We added these two options to both the reference programs and the optimized programs to demonstrate the accuracy of our approach in providing test oracles, without relying on changing the execution options. We can see that in the $iteration_2$, $P_2^{opt}$ produced different results from that of $P_2^{ref}$.

**Table 13: Bug 2 found by queryFuzz.**

| iteration | $P_n^{ref}$ | $Facts_{output\_rel}^{ref}$ | $P_n^{opt}$ | $Facts_{output\_rel}^{opt}$ |
|---|---|---|---|---|
| 0 | `.decl IBf(U:unsigned)`<br>`.decl dfm(U:unsigned)`<br>`.output dfm`<br>`IBf(1).`<br>`dfm(a) :- IBf(a).` | `dfm(1).` | `.decl IBf(U:unsigned)`<br>`.decl dfm(U:unsigned)`<br>`.input IBf`<br>`.output dfm`<br>`dfm(a) :- IBf(a).` | `dfm(1).` |
| 1 | `.decl IBf(U:unsigned)`<br>`.decl dfm(U:unsigned)`<br>`.decl onG(U:unsigned, V:unsigned)`<br>`.output onG`<br>`IBf(1). dfm(1).`<br>`onG(a,b) :- IBf(a), dfm(b), IBf(a).` | `onG(1, 1).` | `.decl IBf(U:unsigned)`<br>`.decl dfm(U:unsigned)`<br>`.decl onG(U:unsigned, V:unsigned)`<br>`.input IBf`<br>`.output onG`<br>`dfm(a) :- IBf(a).`<br>`onG(a,b) :- IBf(a), dfm(b), IBf(a).` | `onG(1, 1).` |
| 2 | `.decl onG(U:unsigned, V:unsigned)`<br>`.decl YVz(U:unsigned)`<br>`.output YVz`<br>`onG(1, 1).`<br>`YVz(a) :- onG(a,a), onG(b,a).` | `YVz(1).` | `.decl IBf(U:unsigned)`<br>`.decl dfm(U:unsigned)`<br>`.decl onG(U:unsigned, V:unsigned)`<br>`.decl YVz(U:unsigned)`<br>`.input IBf`<br>`.output YVz`<br>`dfm(a) :- IBf(a).`<br>`onG(a,b) :- IBf(a), dfm(b), IBf(a).`<br>`YVz(a) :- onG(a,a), onG(b,a).` | - |

*2.1.3 Bug 3.* The bug 3 was caused by the different behaviors of the Datalog engine on a corner case under an optimization. In first rule, there will be `-nan` in the facts of `A`, because there is a negative base (*i.e.,* `-1.5`) and a floating point exponent (*i.e.,* `1.1`). The bug occurred in the second rule. In the default execution, Soufflé thought `-nan` equals to `-nan`, so `-nan` would appear in the `B`'s facts. But when added the `--disable-transformers=ResolveAliasesTransformer` option, Soufflé thought `-nan` not equals to `-nan`, `-nan` would not appear in the `B`'s facts. In Soufflé, we can not take `-nan` and `inf` as input, so we construct `-nan` with $(-1)\hat{}0.5$, construct `inf` with $1/0$. We added the `--disable-transformers=ResolveAliasesTransformer` option for both reference programs and optimized programs and find this bug in two iterations.

**Table 14: Bug 3 found by queryFuzz.**

| iteration | $P_n^{ref}$ | $Facts_{output\_rel}^{ref}$ | $P_n^{opt}$ | $Facts_{output\_rel}^{opt}$ |
|---|---|---|---|---|
| 0 | `.decl fac(U:float, V:float)`<br>`.decl A(U:float)`<br>`.output A`<br>`fac(2.1, -1.5). fac(1.1, 1.1).`<br>`A(n^m) :- fac(v,n), fac(m,m).` | `A(-nan).`<br>`A(1.11053).` | `.decl fac(U:float, V:float)`<br>`.decl A(U:float)`<br>`.output A`<br>`fac(2.1, -1.5). fac(1.1, 1.1).`<br>`A(n^m) :- fac(v,n), fac(m,m).` | `A(-nan).`<br>`A(1.11053).` |
| 1 | `.decl A(U:float)`<br>`.decl B(U:float)`<br>`.output B`<br>`A((-1)^0.5). A(1.11053).`<br>`B(m/m) :- A(m), m = m.` | `B(-nan).`<br>`B(1).` | `.decl fac(U:float, V:float)`<br>`.decl A(U:float)`<br>`.decl B(U:float)`<br>`.output B`<br>`fac(2.1, -1.5). fac(1.1, 1.1).`<br>`A(n^m) :- fac(v,n), fac(m,m).`<br>`B(m/m) :- A(m), m = m.` | `B(1).` |

*2.1.4 Bug 4.* The bug 4 was caused by the Soufflé because it missed transforming some operator (*i.e.*, float-operator) in magic transformation. In this program, there is a relation `Fail` with no attributes. This is a language feature in Soufflé, and it is usually used as a boolean variable. When it equals to `true`, it takes "()" as results, and it has empty results when it equals to `false`. We do not ignore the empty result of this type of relations. We cannot directly construct inputs for this type of relation. In our implementation, we take `Fail():-1=1.` as its input for `true` value, and don't need inputs for `false` value. We show the process that detect this bug with our test oracle generation approach, *i.e.*, evaluate the rules one-by-one. In order to generate this test case, we set a probability for DEOPT so that it allowed the rule with empty results.

**Table 15: Bug 4 found by queryFuzz.**

| iteration | $P_n^{ref}$ | $Facts_{output\_rel}^{ref}$ | $P_n^{opt}$ | $Facts_{output\_rel}^{opt}$ |
|---|---|---|---|---|
| 0 | `.decl Pairs(A:float, B:float)`<br>`.decl First(A:float)`<br>`.output First`<br>`Pairs(0,0).`<br>`First(x) :- Pairs(x,_).` | `First(0).` | `.decl Pairs(A:float, B:float)`<br>`.decl First(A:float)`<br>`.output First`<br>`Pairs(0,0).`<br>`First(x) :- Pairs(x,_).` | `First(0).` |
| 1 | `.decl First(A:float)`<br>`.decl DupFirst(A:float, B:float)`<br>`.output DupFirst`<br>`First(0).`<br>`DupFirst(x,x) :- First(x), x < 100.` | `DupFirst(0, 0).` | `.decl Pairs(A:float, B:float)`<br>`.decl First(A:float)`<br>`.decl DupFirst(A:float, B:float)`<br>`.output DupFirst`<br>`Pairs(0,0).`<br>`First(x) :- Pairs(x,_).`<br>`DupFirst(x,x) :- First(x), x < 100.` | `DupFirst(0, 0).` |
| 2 | `.decl DupFirst(A:float, B:float)`<br>`.decl FirstAgain(A:float)`<br>`.output FirstAgain`<br>`DupFirst(0, 0).`<br>`FirstAgain(x) :- DupFirst(x,_).` | `FirstAgain(0).` | `.decl Pairs(A:float, B:float)`<br>`.decl First(A:float)`<br>`.decl DupFirst(A:float, B:float)`<br>`.decl FirstAgain(A:float)`<br>`.output FirstAgain`<br>`Pairs(0,0).`<br>`First(x) :- Pairs(x,_).`<br>`DupFirst(x,x) :- First(x), x < 100.`<br>`FirstAgain(x) :- DupFirst(x,_).` | `FirstAgain(0).` |
| 3 | `.decl FirstAgain(A:float)`<br>`.decl Fail()`<br>`.output Fail`<br>`FirstAgain(0).`<br>`Fail() :- FirstAgain(x), !FirstAgain(x).` | `Fail(False).` | `.decl Pairs(A:float, B:float)`<br>`.decl First(A:float)`<br>`.decl DupFirst(A:float, B:float)`<br>`.decl FirstAgain(A:float)`<br>`.decl Fail()`<br>`.output Fail`<br>`Pairs(0,0).`<br>`First(x) :- Pairs(x,_).`<br>`DupFirst(x,x) :- First(x), x < 100.`<br>`FirstAgain(x) :- DupFirst(x,_).`<br>`Fail() :- FirstAgain(x), !FirstAgain(x).` | `Fail(False).` |
| 4 | `.decl First(A:float)`<br>`.decl Fail()`<br>`.decl Out(A:float)`<br>`.output Out`<br>`First(0).`<br>`Out(x) :- Fail(), First(x).` | - | `.decl Pairs(A:float, B:float)`<br>`.decl First(A:float)`<br>`.decl DupFirst(A:float, B:float)`<br>`.decl FirstAgain(A:float)`<br>`.decl Fail()`<br>`.decl Out(A:float)`<br>`.output Out`<br>`Pairs(0,0).`<br>`First(x) :- Pairs(x,_).`<br>`DupFirst(x,x) :- First(x), x < 100.`<br>`FirstAgain(x) :- DupFirst(x,_).`<br>`Fail() :- FirstAgain(x), !FirstAgain(x).`<br>`Out(x) :- Fail(), First(x).` | `Out(0).` |

*2.1.5 Bug 5.* The bug 5 was related to the `inline` keyword. Here, we were unable to add inline optimization to the reference program because Soufflé does not allow adding inline optimization for input and output relations.

**Table 16: Bug 5 found by queryFuzz.**

| iteration | $P_n^{ref}$ | $Facts_{output\_rel}^{ref}$ | $P_n^{opt}$ | $Facts_{output\_rel}^{opt}$ |
|---|---|---|---|---|
| 0 | `.decl A(x:float)`<br>`.output A`<br>`A(0.0).`<br>`A(x+1)  :- A(x), x=-0.0.` | `A(0).`<br>`A(1).` | `.decl A(x:float)`<br>`.output A`<br>`A(0.0).`<br>`A(x+1)  :- A(x), x=-0.0.` | `A(0).`<br>`A(1).` |
| 1 | `.decl A(x:float)`<br>`.decl B(A:float, B:float, C:float)`<br>`.output B`<br>`A(0). A(1).`<br>`B(e,e,max(-D,e)) :- A(D), A(e).` | `B(0, 0, -0).`<br>`B(0, 0, 0).`<br>`B(1, 1, 1).` | `.decl A(x:float)`<br>`.decl B(A:float, B:float, C:float)`<br>`.output B`<br>`A(0.0).`<br>`A(x+1)  :- A(x), x=-0.0.`<br>`B(e,e,max(-D,e)) :- A(D), A(e).` | `B(0, 0, -0).`<br>`B(0, 0, 0).`<br>`B(1, 1, 1).` |
| 2 | `.decl A(x:float)`<br>`.decl B(A:float, B:float, C:float)`<br>`.decl out(A:float)`<br>`.output out`<br>`A(0). A(1).`<br>`B(0, 0, -0). B(0, 0, 0). B(1, 1, 1).`<br>`out(R) :- A(R), B(a,b,a), B(a,R,R), B(R,R,R).` | `out(0).`<br>`out(1).` | `.decl A(x:float)`<br>`.decl B(A:float, B:float, C:float) inline`<br>`.decl out(A:float)`<br>`.output out`<br>`A(0.0).`<br>`A(x+1)  :- A(x), x=-0.0.`<br>`B(e,e,max(-D,e)) :- A(D), A(e).`<br>`out(R) :- A(R), B(a,b,a), B(a,R,R), B(R,R,R).` | `out(-0).`<br>`out(0).`<br>`out(1).` |

*2.1.6 Bug 6.* The bug 6 was caused by the data structure `brie` and only occurred in synthesizer mode. `brie` is also an optimization method. In this example, we evaluate the optimized program with the synthesizer mode of Soufflé and found this bug with our approach. To demonstrate that our test oracles generation method did mitigate the bugs caused by optimization, we evaluated the reference program with synthesizer mode and add `brie` to $P_4^{ref}$, and the result showed that our reference program can be executed correctly.

**Table 17: Bug 6 found by queryFuzz.**

| iteration | $P_n^{ref}$ | $Facts_{output\_rel}^{ref}$ | $P_n^{opt}$ | $Facts_{output\_rel}^{opt}$ |
|---|---|---|---|---|
| 0 | `.decl BCCS(A:number, B:number)`<br>`.decl Bkft(A:number)`<br>`.output Bkft`<br>`BCCS(12, 4). BCCS(1, 15).`<br>`Bkft(max(--Q,-9)) :- BCCS(IAV,Q).` | `Bkft(4).`<br>`Bkft(15).` | `.decl UteE(A:number, B:number)`<br>`.decl BCCS(A:number, B:number)`<br>`.decl Bkft(A:number)`<br>`.output Bkft`<br>`UteE(-6, 15). UteE(5, 4).`<br>`UteE(-10, 4). UteE(-10, -29).`<br>`BCCS(12, 4). BCCS(1, 15).`<br>`Bkft(max(--Q,-9)) :- BCCS(IAV,Q).` | `Bkft(4).`<br>`Bkft(15).` |
| 1 | `.decl BCCS(A:number, B:number)`<br>`.decl HXxe(A:number)`<br>`.output HXxe`<br>`BCCS(12, 4). BCCS(1, 15).`<br>`HXxe(S-5-9*Oiz) :- BCCS(S,Oiz).` | `HXxe(-139).`<br>`HXxe(-29).` | `.decl UteE(A:number, B:number)`<br>`.decl BCCS(A:number, B:number)`<br>`.decl Bkft(A:number)`<br>`.decl HXxe(A:number)`<br>`.output HXxe`<br>`UteE(-6, 15). UteE(5, 4).`<br>`UteE(-10, 4). UteE(-10, -29).`<br>`BCCS(12, 4). BCCS(1, 15).`<br>`Bkft(max(--Q,-9)) :- BCCS(IAV,Q).`<br>`HXxe(S-5-9*Oiz) :- BCCS(S,Oiz).` | `HXxe(-139).`<br>`HXxe(-29).` |
| 2 | `.decl UteE(A:number, B:number)`<br>`.decl HXxe(A:number)`<br>`.decl Appy(A:number)`<br>`.output Appy`<br>`UteE(-6, 15). UteE(5, 4).`<br>`UteE(-10, 4). UteE(-10, -29).`<br>`HXxe(-139). HXxe(-29).`<br>`Appy(D) :- HXxe(D), UteE(SNu,D).` | `Appy(-29).` | `.decl UteE(A:number, B:number)`<br>`.decl BCCS(A:number, B:number)`<br>`.decl Bkft(A:number)`<br>`.decl HXxe(A:number)`<br>`.decl Appy(A:number)`<br>`.output Appy`<br>`UteE(-6, 15). UteE(5, 4).`<br>`UteE(-10, 4). UteE(-10, -29).`<br>`BCCS(12, 4). BCCS(1, 15).`<br>`Bkft(max(--Q,-9)) :- BCCS(IAV,Q).`<br>`HXxe(S-5-9*Oiz) :- BCCS(S,Oiz).`<br>`Appy(D) :- HXxe(D), UteE(SNu,D).` | `Appy(-29).` |
| 3 | `.decl BCCS(A:number, B:number)`<br>`.decl Bkft(A:number)`<br>`.decl Appy(A:number)`<br>`.output Bkft`<br>`BCCS(12, 4). BCCS(1, 15).`<br>`Appy(-29).`<br>`Bkft(Y) :- BCCS(rzG,Y), !Appy(Y).` | `Bkft(4).`<br>`Bkft(15).` | `.decl UteE(A:number, B:number)`<br>`.decl BCCS(A:number, B:number)`<br>`.decl Bkft(A:number)`<br>`.decl HXxe(A:number)`<br>`.decl Appy(A:number)`<br>`.output Bkft`<br>`UteE(-6, 15). UteE(5, 4).`<br>`UteE(-10, 4). UteE(-10, -29).`<br>`BCCS(12, 4). BCCS(1, 15).`<br>`Bkft(max(--Q,-9)) :- BCCS(IAV,Q).`<br>`HXxe(S-5-9*Oiz) :- BCCS(S,Oiz).`<br>`Appy(D) :- HXxe(D), UteE(SNu,D).`<br>`Bkft(Y) :- BCCS(rzG,Y), !Appy(Y).` | `Bkft(4, 15).` |
| 4 | `.decl UteE(A:number, B:number)`<br>`.decl Bkft(A:number)`<br>`.decl bNtR(A:number) brie`<br>`.output bNtR`<br>`UteE(-6, 15). UteE(5, 4).`<br>`UteE(-10, 4). UteE(-10, -29).`<br>`Bkft(4). Bkft(15).`<br>`bNtR(J) :- UteE(J,VHk), Bkft(VHk).` | `bNtR(5).`<br>`bNtR(-6).`<br>`bNtR(-10).` | `.decl UteE(A:number, B:number)`<br>`.decl BCCS(A:number, B:number)`<br>`.decl Bkft(A:number)`<br>`.decl HXxe(A:number)`<br>`.decl Appy(A:number)`<br>`.decl bNtR(A:number) brie`<br>`.output bNtR`<br>`UteE(-6, 15). UteE(5, 4).`<br>`UteE(-10, 4). UteE(-10, -29).`<br>`BCCS(12, 4). BCCS(1, 15).`<br>`Bkft(max(--Q,-9)) :- BCCS(IAV,Q).`<br>`HXxe(S-5-9*Oiz) :- BCCS(S,Oiz).`<br>`Appy(D) :- HXxe(D), UteE(SNu,D).`<br>`Bkft(Y) :- BCCS(rzG,Y), !Appy(Y).`<br>`bNtR(J) :- UteE(J,VHk), Bkft(VHk).` | `bNtR(5).`<br>`bNtR(-10).` |

*2.1.7 Bug 7.* Similar with the bug 6, the bug 7 was also caused by the data structure `brie` and only occurred in synthesizer mode. It was challenging for us to generate test oracles for this test case because it only contained a single rule. To overcome this issue, we made an equivalent transformation to add a new rule without modifying its semantics, as shown in Figure 3. Unlike the previous test cases, we were able to trigger this bug using the reference program, which allowed us to identify it.

```
.decl wRnH(A:float, B:float)
.decl out(A:float) brie

.output out
wRnH(-21.360, -21.360). wRnH(-20.607, 2.486).
wRnH(2.486, 2.486). wRnH(-13.652, -13.652).
wRnH(-18.915, -18.915).

out(r) :- wRnH(a, r), wRnH(r, r).
```

**(a) The original test case.**

```
.decl wRnH(A:float, B:float)
.decl out(A:float) brie inline
.decl res(A:float)
.output res
wRnH(-21.360, -21.360). wRnH(-20.607, 2.486).
wRnH(2.486, 2.486). wRnH(-13.652, -13.652).
wRnH(-18.915, -18.915).
out(r) :- wRnH(a, r), wRnH(r, r).
res(a) :- out(a).
```

**(b) The transformed test case.**

**Figure 3: An equivalent transformation for bug 7.**

**Table 18: Bug 7 found by queryFuzz.**

| iteration | $P_n^{ref}$ | $Facts_{output\_rel}^{ref}$ | $P_n^{opt}$ | $Facts_{output\_rel}^{opt}$ |
|---|---|---|---|---|
| 0 | `.decl wRnH(A:float, B:float)`<br>`.decl out(A:float) brie`<br>`.output out`<br>`wRnH(-21.360, -21.360).`<br>`wRnH(-20.607, 2.486).`<br>`wRnH(2.486, 2.486).`<br>`wRnH(-13.652, -13.652).`<br>`wRnH(-18.915, -18.915).`<br>`out(r) :- wRnH(a,r), wRnH(r,r).` | `out(-18.9150009).`<br>`out(-13.6520004).`<br>`out(2.48600006).` | `.decl wRnH(A:float, B:float)`<br>`.decl out(A:float) brie`<br>`.output out`<br>`wRnH(-21.360, -21.360).`<br>`wRnH(-20.607, 2.486).`<br>`wRnH(2.486, 2.486).`<br>`wRnH(-13.652, -13.652).`<br>`wRnH(-18.915, -18.915).`<br>`out(r) :- wRnH(a,r), wRnH(r,r).` | `out(-18.9150009).`<br>`out(-13.6520004).`<br>`out(2.48600006).` |
| 1 | `.decl out(A:float) brie`<br>`.decl res(A:float)`<br>`.output res`<br>`out(-18.9150009).`<br>`out(-13.6520004).`<br>`out(2.48600006).`<br>`res(a) :- out(a).` | `res(-18.9150009).`<br>`res(-13.6520004).`<br>`res(2.48600006).` | `.decl wRnH(A:float, B:float)`<br>`.decl out(A:float) brie inline`<br>`.decl res(A:float)`<br>`.output res`<br>`wRnH(-21.360, -21.360).`<br>`wRnH(-20.607, 2.486).`<br>`wRnH(2.486, 2.486).`<br>`wRnH(-13.652, -13.652).`<br>`wRnH(-18.915, -18.915).`<br>`out(r) :- wRnH(a,r), wRnH(r,r).`<br>`res(a) :- out(a).` | `res(-21.3600006).`<br>`res(-18.9150009).`<br>`res(-13.6520004).`<br>`res(2.48600006).` |

*2.1.8 Bug 8.* Similar to bug 7, bug 8 also occurred within a single rule, but the difference was that the rule in bug 8 was not influenced by other rules. This bug is introduced by the optimization between two subgoals in a rule. It was difficult for us to generate test oracles for this bug because it did not align with our expectations, which were based on the assumption that the bug was introduced by the optimization between different rules. Therefore, it was not possible to detect this bug by using the same options in the reference program and the optimized program. However, we were able to identify it by removing the execution options in the reference program. This bug was only triggered when execute the program with the option `--disable-transformers=ResolveAliasesTransformer`.

```
.type Word <: symbol
.decl words (i:number, w:Word)
.decl out (w:Word, w1:Word)
.output out
words(0,"a").
words(1, "b").
out(w, w1)  :- words(i, w), words(i+1, w1).

// expected result:
// out("a", "b").
// buggy result:
// out("a", "a"). out("a", "b"). out("b", "a"). out("b", "b").
```

**Figure 4: Bug 8 found by queryFuzz.**

*2.1.9 Bug 9.* Bug 9 was a bug in μZ and also occurred in only one rule. We made an equivalent transformation for this program and were able to trigger the bug in the reference program.

```
in(A:Z) input
out(A:Z) printtuples

in(43). in(33).

out(a) :- in(a), in(a), in(a).
```

**(a) The original test case.**

```
aicl(A:Z) input
in(A:Z)
out(A:Z) printtuples
aicl(43). aicl(33).
in(a) :- aicl(a).
out(a) :- in(a), in(a), in(a).
```

**(b) The transformed test case.**

**Figure 5: An equivalent transformation for bug 9.**

**Table 19: Bug 9 found by queryFuzz.**

| iteration | $P_n^{ref}$ | $Facts_{output\_rel}^{ref}$ | $P_n^{opt}$ | $Facts_{output\_rel}^{opt}$ |
|---|---|---|---|---|
| 0 | `Z 64`<br>`aicl(A:Z) input`<br>`in(A:Z) printtuples`<br>`aicl(43). aicl(33).`<br>`in(a) :- aicl(a).` | `in(33).`<br>`in(43).` | `Z 64`<br>`aicl(A:Z) input`<br>`in(A:Z) printtuples`<br>`aicl(43). aicl(33).`<br>`in(a) :- aicl(a).` | `in(33).`<br>`in(43).` |
| 1 | `Z 64`<br>`in(A:Z) input`<br>`out(A:Z) printtuples`<br>`in(43). in(33).`<br>`out(a) :- in(a), in(a), in(a).` | `out(43).`<br>`out(33).`<br>`out(2).`<br>`...`<br>`out(63).` | `Z 64`<br>`aicl(A:Z) input`<br>`in(A:Z)`<br>`out(A:Z) printtuples`<br>`aicl(43). aicl(33).`<br>`in(a) :- aicl(a).`<br>`out(a) :- in(a), in(a), in(a).` | `out(43).`<br>`out(33).` |

*2.1.10 Bug 10.* Bug 10 was a bug in μZ, our method could generate test oracle for this bug.

**Table 20: Bug 10 found by queryFuzz.**

| iteration | $P_n^{ref}$ | $Facts_{output\_rel}^{ref}$ | $P_n^{opt}$ | $Facts_{output\_rel}^{opt}$ |
|---|---|---|---|---|
| 0 | `Z 64`<br>`in1(A:Z) input`<br>`in2(A:Z, B:Z) input`<br>`R(A:Z, B:Z) printtuples`<br>`in1(49). in1(10). in2(25, 10).`<br>`in2(16, 13). in2(24, 22).`<br>`R(V,M) :- in2(V,M), in1(M).` | `R(25, 10).` | `Z 64`<br>`in1(A:Z) input`<br>`in2(A:Z, B:Z) input`<br>`R(A:Z, B:Z) printtuples`<br>`in1(49). in1(10). in2(25, 10).`<br>`in2(16, 13). in2(24, 22).`<br>`R(V,M) :- in2(V,M), in1(M).` | `R(25, 10).` |
| 1 | `Z 64`<br>`R(A:Z, B:Z) input`<br>`out(A:Z) printtuples`<br>`R(25,10).`<br>`out(f) :- R(f,c), R(f,a), R(f,b).` | `out(25).` | `Z 64`<br>`in1(A:Z) input`<br>`in2(A:Z, B:Z) input`<br>`R(A:Z, B:Z)`<br>`out(A:Z) printtuples`<br>`in1(49). in1(10). in2(25, 10).`<br>`in2(16, 13). in2(24, 22).`<br>`R(V,M) :- in2(V,M), in1(M).`<br>`out(f) :- R(f,c), R(f,a), R(f,b).` | `out(49).`<br>`out(10).`<br>`out(25).`<br>`...`<br>`out(63).` |

*2.1.11 Bug 11.* Bug 11 was a bug in μZ and much similar with bug 10, our method could generate test oracle for this bug.

**Table 21: Bug 10 found by queryFuzz.**

| iteration | $P_n^{ref}$ | $Facts_{output\_rel}^{ref}$ | $P_n^{opt}$ | $Facts_{output\_rel}^{opt}$ |
|---|---|---|---|---|
| 0 | `Z 64`<br>`in(A:Z, B:Z) input`<br>`cond(A:Z, B:Z) printtuples`<br>`in(45, 40).`<br>`cond(c,Z) :- in(c,Z), c > Z.` | `cond(45, 40).` | `Z 64`<br>`in(A:Z, B:Z) input`<br>`cond(A:Z, B:Z) printtuples`<br>`in(45, 40).`<br>`cond(c,Z) :- in(c,Z), c > Z.` | `cond(45, 40).` |
| 1 | `Z 64`<br>`in(A:Z, B:Z) input`<br>`cond(A:Z, B:Z) input`<br>`out(A:Z) printtuples`<br>`in(45, 40). cond(45, 40).`<br>`out(y) :- in(A,y), cond(B,y).` | `out(40).` | `Z 64`<br>`in(A:Z, B:Z) input`<br>`cond(A:Z, B:Z)`<br>`out(A:Z) printtuples`<br>`in(45, 40).`<br>`cond(c,Z) :- in(c,Z), c > Z.`<br>`out(y) :- in(A,y), cond(B,y).` | - |

*2.1.12 Bug 12.* Bug 12 was a bug in μZ, and simialr with bug 4, it depended on an empty relation (*i.e.,* `y`). We could generate this test case and generate test oracles fot it. In this test case, the last rule of `out` gave the error result under the influence of unsafe negation in the second rule (*i.e.,* `!WWW (_)`). Our test oracle generation method can block the propagation of this error.

**Table 22: Bug 12 found by queryFuzz.**

| iteration | $P_n^{ref}$ | $Facts_{output\_rel}^{ref}$ | $P_n^{opt}$ | $Facts_{output\_rel}^{opt}$ |
|---|---|---|---|---|
| 0 | `Z 64`<br>`WWW(A:Z) input`<br>`mAqE(A:Z) input`<br>`p(A:Z) printtuples`<br>`WWW(18). WWW(15). WWW(25). WWW(16).`<br>`mAqE(29). mAqE(39).`<br>`p(g) :- mAqE(x), WWW(g), x < g.` | - | `Z 64`<br>`WWW(A:Z) input`<br>`mAqE(A:Z) input`<br>`p(A:Z) printtuples`<br>`WWW(18). WWW(15). WWW(25). WWW(16).`<br>`mAqE(29). mAqE(39).`<br>`p(g) :- mAqE(x), WWW(g), x < g.` | - |
| 1 | `Z 64`<br>`WWW(A:Z) input`<br>`mAqE(A:Z) input`<br>`p(A:Z) input`<br>`y(A:Z) printtuples`<br>`WWW(18). WWW(15). WWW(25). WWW(16).`<br>`mAqE(29). mAqE(39).`<br>`y(d) :- mAqE(a), p(d), !WWW(_).` | - | `Z 64`<br>`WWW(A:Z) input`<br>`mAqE(A:Z) input`<br>`p(A:Z)`<br>`y(A:Z) printtuples`<br>`WWW(18). WWW(15). WWW(25). WWW(16).`<br>`mAqE(29). mAqE(39).`<br>`p(g) :- mAqE(x), WWW(g), x < g.`<br>`y(d) :- mAqE(a), p(d), !WWW(_).` | - |
| 2 | `Z 64`<br>`WWW(A:Z) input`<br>`y(A:Z) input`<br>`out(A:Z) printtuples`<br>`WWW(18). WWW(15). WWW(25). WWW(16).`<br>`out(c) :- y(a), WWW(b), WWW(c).` | - | `Z 64`<br>`WWW(A:Z) input`<br>`mAqE(A:Z) input`<br>`p(A:Z)`<br>`y(A:Z)`<br>`out(A:Z) printtuples`<br>`WWW(18). WWW(15). WWW(25). WWW(16).`<br>`mAqE(29). mAqE(39).`<br>`p(g) :- mAqE(x), WWW(g), x < g.`<br>`y(d) :- mAqE(a), p(d), !WWW(_).`<br>`out(c) :- y(a), WWW(b), WWW(c).` | `out(18).`<br>`out(15).`<br>`out(25).`<br>`out(16).` |

*2.1.13 Bug 13.* Bug 13 was a bug in DDlog, this bug has not been fixed until now and we have detected this bug with our approach. We show the process of generating the oracle for this bug-inducing test case reported by queryFuzz.

**Table 23: Bug 13 found by queryFuzz.**

| iteration | $P_n^{ref}$ | $Facts_{output\_rel}^{ref}$ | $P_n^{opt}$ | $Facts_{output\_rel}^{opt}$ |
|---|---|---|---|---|
| 0 | `input relation Z(a:string)`<br>`output relation PQRI(a:string)`<br>`Z("3u37ezDdBg"). Z("uwsUc").`<br>`Z("zDdBgEbgMd"). Z("EbgMd2DV3K").`<br>`PQRI(v) :- Z(v), Z(nbj).` | `PQRI("3u37ezDdBg").`<br>`PQRI("uwsUc").`<br>`PQRI("zDdBgEbgMd").`<br>`PQRI("EbgMd2DV3K").` | `input relation Z(a:string)`<br>`output relation PQRI(a:string)`<br>`Z("3u37ezDdBg"). Z("uwsUc").`<br>`Z("zDdBgEbgMd"). Z("EbgMd2DV3K").`<br>`PQRI(v) :- Z(v), Z(nbj).` | `PQRI("3u37ezDdBg").`<br>`PQRI("uwsUc").`<br>`PQRI("zDdBgEbgMd").`<br>`PQRI("EbgMd2DV3K").` |
| 1 | `input relation Z(a:string)`<br>`input relation PQRI(a:string)`<br>`output relation PLEY(a:string)`<br>`Z("3u37ezDdBg"). Z("uwsUc").`<br>`Z("zDdBgEbgMd"). Z("EbgMd2DV3K").`<br>`PQRI("3u37ezDdBg"). PQRI("uwsUc").`<br>`PQRI("zDdBgEbgMd"). PQRI("EbgMd2DV3K").`<br>`PLEY(o) :- PQRI(x), Z(o), Z(x),`<br>`PQRI(z), PQRI(x).` | `PLEY("3u37ezDdBg").`<br>`PLEY("uwsUc").`<br>`PLEY("zDdBgEbgMd").`<br>`PLEY("EbgMd2DV3K").` | `input relation Z(a:string)`<br>`relation PQRI(a:string)`<br>`output relation PLEY(a:string)`<br>`Z("3u37ezDdBg"). Z("uwsUc").`<br>`Z("zDdBgEbgMd"). Z("EbgMd2DV3K").`<br>`PQRI(v) :- Z(v), Z(nbj).`<br>`PLEY(o) :- PQRI(x), Z(o), Z(x),`<br>`PQRI(z), PQRI(x).` | `PLEY("3u37ezDdBg").`<br>`PLEY("uwsUc").`<br>`PLEY("zDdBgEbgMd").`<br>`PLEY("EbgMd2DV3K").` |
| 2 | `input relation Z(a:string)`<br>`input relation PLEY(a:string)`<br>`output relation NFUV(a:string)`<br>`Z("3u37ezDdBg"). Z("uwsUc").`<br>`Z("zDdBgEbgMd"). Z("EbgMd2DV3K").`<br>`PLEY("3u37ezDdBg"). PLEY("uwsUc").`<br>`PLEY("zDdBgEbgMd"). PLEY("EbgMd2DV3K").`<br>`NFUV(q) :- Z(fym), PLEY(q).` | `NFUV("3u37ezDdBg").`<br>`NFUV("uwsUc").`<br>`NFUV("zDdBgEbgMd").`<br>`NFUV("EbgMd2DV3K").` | `input relation Z(a:string)`<br>`relation PQRI(a:string)`<br>`relation PLEY(a:string)`<br>`output relation NFUV(a:string)`<br>`Z("3u37ezDdBg"). Z("uwsUc").`<br>`Z("zDdBgEbgMd"). Z("EbgMd2DV3K").`<br>`PQRI(v) :- Z(v), Z(nbj).`<br>`PLEY(o) :- PQRI(x), Z(o), Z(x),`<br>`PQRI(z), PQRI(x).`<br>`NFUV(q) :- Z(fym), PLEY(q).` | `NFUV("3u37ezDdBg").`<br>`NFUV("uwsUc").`<br>`NFUV("zDdBgEbgMd").`<br>`NFUV("EbgMd2DV3K").` |
| 3 | `input relation PLEY(a:string)`<br>`input relation NFUV(a:string)`<br>`output relation OUT(a:string)`<br>`NFUV("3u37ezDdBg"). NFUV("uwsUc").`<br>`NFUV("zDdBgEbgMd"). NFUV("EbgMd2DV3K").`<br>`PLEY("3u37ezDdBg"). PLEY("uwsUc").`<br>`PLEY("zDdBgEbgMd"). PLEY("EbgMd2DV3K").`<br>`OUT(t) :- NFUV(ssz), PLEY(arv), PLEY(t).` | `OUT("3u37ezDdBg").`<br>`OUT("uwsUc").`<br>`OUT("zDdBgEbgMd").`<br>`OUT("EbgMd2DV3K").` | `input relation Z(a:string)`<br>`relation PQRI(a:string)`<br>`relation PLEY(a:string)`<br>`relation NFUV(a:string)`<br>`output relation OUT(a:string)`<br>`Z("3u37ezDdBg"). Z("uwsUc").`<br>`Z("zDdBgEbgMd"). Z("EbgMd2DV3K").`<br>`PQRI(v) :- Z(v), Z(nbj).`<br>`PLEY(o) :- PQRI(x), Z(o), Z(x),`<br>`PQRI(z), PQRI(x).`<br>`NFUV(q) :- Z(fym), PLEY(q).`<br>`OUT(t) :- NFUV(ssz), PLEY(arv), PLEY(t).` | - |

## 2.2 Bugs found by Deopt

There were 5 bugs might not be detectable by queryFuzz: bug 3 (*i.e.,* shown in Section 1.3), bug 4 (*i.e.,* shown in Section 1.4), bug 5 (shown in Section 1.5), bug 8 (*i.e.,* shown in Section 1.8), and bug 13 (*i.e.,* shown in Section 1.13). queryFuzz's method for detecting bugs in programs beyond conjunctive queries relies on the concept of monotonicity; in short, queryFuzz applies transformations only to conjunctive queries, and the final results should not contradict their oracles, because of monotonicity.

Bug 3 was caused by subsumption, which removes facts from a relation based on the conditions specified in its body. If there are no facts that meet these conditions, the subsumption will not remove any facts. This bug causes the subsumption to fail to remove any facts, even if they meet the specified conditions. This behavior is identical to the case where no facts meet the conditions, as queryFuzz is unaware of whether any facts meet the conditions, it was impossible for queryFuzz to detect it. Bug 4 and bug 8 were also difficult for queryFuzz to detect for the same reason.

Bug 5 was related to equivalence relations, and it not works under magic transformation. Equivalence relation is a binary relation in Soufflé and exhibits three properties: reflexivity, symmetry, and transitivity. For example, deriving a fact `(1, 2)` for an equivalence relation `c`, will result in the following results: `c(1, 2). c(2, 1). c(1, 1). c(2, 2).`. So when given an input to the equivalence relation, that input should be a subset of the final result, but there is no way to tell if they are equal (*e.g.,*, if the fact derived for `c` is `(1, 1)`, the results of `c` will be `c(1, 1).`). So it is challenging for queryFuzz to determine whether the equivalence relation works well, resulting in its inability to detect this bug.

Bug 13 was a bug in CozoDB, which was caused by the discrepancy between the comparison functions employed in the magic sets rewrite and the binary operators utilized in the rules. During the magic sets rewrite, CozoDB employs range scanning on the relation `a` instead of testing each individual value. However, the comparison function used in the range scan considers `-0.0 >= 0` to be true, while the comparison function used in the binary operators considers it to be false. This bug can only be identified when the program is evaluated twice, once with

the magic sets rewrite and once without it. We believe that queryFuzz would not be able to detect this bug because the transformations it applies are designed specifically for conjunctive queries, which consist of single non-recursive function-free Horn rules. We got feedback from the developer of CozoDB that their magic sets rewrite only applied on conjunctive queries. Therefore, the transformations supported by queryFuzz, such as adding an existing subgoal into the rule body, modifying a variable, or removing a subgoal from the rule body, are inadequate for preventing the magic sets rewrite in this particular scenario.