A Variables and Constants in Interpolants

In addition to the number of logical connectives in an interpolant, we also report the number of distinct variables and the largest constant appearing in each interpolant.

Table 4: Variables and constants. Numbers are averaged over all valid solutions across all test problems.

Model	**trained with Z3 GT**		**trained with CSIsat GT**		**Original**	
	Variables	Constants	Variables	Constants	Variables	Constants
Z3	2.45	5.97	2.39	6.39	2.45	4.24
CSIsat	2.50	16386579	2.52	460537	2.60	7476237725

The number of variables in a solution does not change significantly when either solver is guided by the model. However, the largest constants in CSIsat-generated solutions drop by several orders of magnitude when the solver is guided by the model. This may be attributed to CSIsat lacking effective heuristics for optimizing over integer domains.

B Example Problems from Dataset

Below we include three example problems from **Interp**(4,4). The included "Z3-solutions" are solutions generated by the original Z3.

The "simp-solutions" are the solutions found by Z3 when guided by the model trained specifically for Z3. These are not part of the original **Interp**$(4, 4)$ dataset, but we include them here for reference.

```
{
  'input1': [
    '1 + v_sep_x2 <= v_sep_x3',
    '(v_sep_x3 <= v_sep_x1 and (6 == v_sep_x0 and 1 + v_sep_x0 <=
       v_sep_x3))',
    '(95 <= v_sep_x1 and v_sep_x1 <= 100)',
    '(1 <= v_sep_x0 and 1 + v_sep_x0 <= v_sep_x1)'
  ],

  'input2': [
    '(1 == v_sep_x1 and (0 <= v_sep_x3 and (v_sep_x1 <= v_sep_x2 and 1
        + v_sep_x3 <= v_sep_x2)))',
    '(0 == v_sep_x1 and (v_sep_x2 == v_sep_x3 and 1 <= v_sep_x3))',
    '(1 + v_sep_x3 <= 0 and (1 == v_sep_x1 and (v_sep_x1 == v_sep_x2
       and 1 <= v_sep_x2)))'
  ],

  'Z3-solution': '( not (v_sep_x1  <=  1)  ||  not (v_sep_x3 + -1*
     v_sep_x2  <=  -1))  && ( not (v_sep_x1  <=  0)  ||  not (
     v_sep_x3 + -1*v_sep_x2  <=  0))',

  'simp-solution': ' not (v_sep_x1  <=  1)  ||  not (v_sep_x2 + -1*
     v_sep_x3  >=  0)',

  'time': 0.012347221374511719
}


 {
   'input1': [
     '1 + v_sep_x2 <= v_sep_x3',
     '(92 <= v_sep_x1 and v_sep_x1 <= 97)',
     '(2 + v_sep_x2 <= v_sep_var961 and (3 + v_sep_x2 <= v_sep_var961
        and (2 == v_sep_var961 and (v_sep_var961 <= v_sep_var962 and
        v_sep_var961 <= v_sep_var973))))',
```

```
    '(2 + v_sep_x2 <= v_sep_var702 and (4 + v_sep_x2 <= v_sep_var702
       and (3 == v_sep_var702 and (v_sep_var702 <= v_sep_var703 and
       v_sep_var702 <= v_sep_var711))))'
  ],

  'input2': [
    '(0 == v_sep_x3 and (0 == 1 + v_sep_var3004 and (1 ==
       v_sep_var3005 and (2 == v_sep_x1 and 2 <= v_sep_x2))))',
    '(1 == v_sep_x1 and (0 <= v_sep_x3 and (v_sep_x1 <= v_sep_x2 and 1
       + v_sep_x3 <= v_sep_x2)))',
    '(1 + v_sep_x3 <= 0 and (1 == v_sep_x1 and (v_sep_x1 == v_sep_x2
       and 1 <= v_sep_x2)))',
    '(0 == v_sep_x1 and (v_sep_x2 == v_sep_x3 and 1 <= v_sep_x3))'
  ],

  'Z3-solution': '( not (v_sep_x1 <= 1) || not (v_sep_x3 >= 0)
      || not (v_sep_x2 + -1*v_sep_x3 >= 0)) && ( not (v_sep_x1
     <= 1) || not (v_sep_x3 <= 1) || not (v_sep_x2 >= 1))
      && ( not (v_sep_x1 <= 2) || not (v_sep_x3 <= 0) ||
     not (v_sep_x2 >= 2))',

  'simp-solution': '( not (v_sep_x1 <= 2) || not (v_sep_x2 >=
     0) || not (v_sep_x3 <= 0)) && ( not (v_sep_x1 <= 1) ||
      not (v_sep_x3 >= 0) || not (v_sep_x3 + -1*v_sep_x2 <= 0))
     ',

  'time': 0.014998197555541992
 }


{
  'input1': [
    'v_sep_x3 == 1 + v_sep_x2',
    '(96 <= v_sep_x1 and v_sep_x1 <= 100)',
    '(92 <= v_sep_x1 and v_sep_var122 <= 100)',
    '(1 + v_sep_x2 <= 0 and (1 + v_sep_x3 <= 0 and v_sep_x3 <= 1))'
  ],

  'input2': [
    '(0 == v_sep_x3 and (0 == 1 + v_sep_var3004 and (1 ==
       v_sep_var3005 and (2 == v_sep_x1 and 2 <= v_sep_x2))))',
    '(1 == v_sep_x1 and (0 <= v_sep_x3 and (v_sep_x1 <= v_sep_x2 and 1
       + v_sep_x3 <= v_sep_x2)))',
    '(1 + v_sep_x3 <= 0 and (1 == v_sep_x1 and (v_sep_x1 == v_sep_x2
       and 1 <= v_sep_x2)))',
    '(0 == v_sep_x1 and (v_sep_x2 == v_sep_x3 and 1 <= v_sep_x3))'
  ],

  'Z3-solution': '( not (v_sep_x1 <= 2) || not (v_sep_x3 <= 0)
      || not (v_sep_x2 >= 2)) && ( not (v_sep_x1 <= 1) ||
     not (v_sep_x3 <= 1) || not (v_sep_x2 >= 1)) && ( not (
     v_sep_x1 <= 1) || not (v_sep_x3 >= 0) || not (v_sep_x3
      + -1*v_sep_x2 <= 0))',

  'simp-solution': '( not (v_sep_x1 <= 1) || not (v_sep_x3 >= 0)
      || not (v_sep_x2 + -1*v_sep_x3 >= 0)) && ( not (v_sep_x1
     <= 2) || not (v_sep_x3 <= 0) || not (v_sep_x2 >= 1))'
     ,

  'time': 0.014645099639892578
}
```

C Experiments with CVC5

In addition to Z3 and CSIsat, we also report a limited evaluation with CVC5 [3]. Technically, while CVC5 supports interpolation, it struggles with our benchmark suite. In our experiments, CVC5 failed to solve 85% of the problems within the timeout of 5 seconds. For the remaining 15%, the problems were so simple that the original interpolants already had complexity 1 and could not be further simplified.

Nonetheless, we observe a significant reduction (-28% with a conservative estimation) in solver time which is comparable to the 30% reduction observed for Z3 in Table 2. Table 5 shows the statistics on guiding CVC5 using the model trained on Z3's ground truths.

Table 5: Solver performance (mean $\pm$ 95% CI) for solver time.

Method	**Valid**	**Complexity**	**Solver time (s)**
Original CVC5	15.3%	1	0.301 ± 0.116
Guided CVC5	7.3%	1	0.107 ± 0.031