Relational Program Synthesis with Numerical Reasoning

UK Research and Innovation

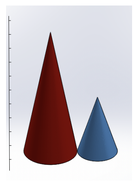
Céline Hocquette and Andrew Cropper

Departement of Computer Science
University of Oxford
celine.hocquette@cs.ox.ac.uk

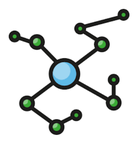
UNIVERSITY OF OXFORD

1 - Introduction

Learning programs with numerical values is fundamental to many applications.



```
zendo(Game) ← piece(Game,Piece1),
              contact(Piece1,Piece2),
              size(Piece2,Size),geq(Size,7).
```



```
pharma3(Mol) ← zincsite(Mol,Zinc),
               hacc(Mol,Hydro),
               dist(Mol,Zinc,Hydro,Dist),
               leq(Dist,3.58),geq(D,1.78).
pharma3(Mol) ← hacc(Mol,Hydro1),
               hacc(Mol,Hydro2),
               dist(Mol,Hydro1,Hydro2,Dist),
               leq(Dist,2.78),
               bond(Mol,Hydro1,Hydro2,du).
```

Fig. 1: Example of programs with numerical values.

Existing program synthesis approaches:

1. struggle to identify numerical values from infinite domains [2, 1]
2. cannot perform complex numerical reasoning such as reasoning over multiple examples jointly [4, 3]

We introduce an approach, implemented in NUMSYNTH, which combines relational learning and numerical reasoning to efficiently learn programs with numerical values.

3 - Implementation

NUMSYNTH uses a set of built-in numerical literals (Table 1) to support a large class of arithmetical fragments (Table 2).

Literal	Definition	Example
`geq(A,N)`	$A \geq N$	`geq(A,3)`
`leq(A,N)`	$A \leq N$	`leq(A,5.2)`
`add(A,B,C)`	$A + B = C$	`add(A,B,C)`
`mult(A,N,C)`	$A * N = C$	`mult(A,2,C)`

Table 1: Numerical literals in NUMSYNTH. N is a numerical variable. A, B, C, N are real numbers or integers.

Fragment	NUMSYNTH	Example
Linear real arithm.	✓	$X + 6.3 * Y \leq 3$
Linear integer arithm.	✓	$U + 6 * V \leq 3$
Mixed real / integer	✓	$X + 6.3 * V \leq 3$
Integer difference logic	✓	$U - V \leq 4$
Real difference logic	✓	$X - Y \leq 4$
Unit two-variable ineq.	✓	$X + Y \leq 4$
Polynomial real arithm.	X	$X^2 + Y^2 = 2$
Nonlinear integer arithm.	X	$U^2 = 2$

Table 2: Arithmetical fragments supported by NUMSYNTH. X and Y are real numbers and U and V integers.

2 - Our approach

Key idea: separate the search in two stages. Inspired by ALEPH's lazy evaluation procedure [5].

1. **program search**: generate partial hypotheses with *numerical variables* in place of numerical symbols: `f(List) ← length(List,Length), leq(Length,N), @numerical(N)`
2. **numerical search**: searches for numerical values to fill in the numerical variables.
   (a) finds values for the intermediate variable Length given the positive and negative examples:

$$E^+ = \{f([a,b,c,d,e]), f([l,n,t,n])\} \text{ and } E^- = \{f([u,v,w]), f([r,s])\}$$
$$S_{E^+}(Length) = \{5,4\} \text{ and } S_{E^-}(Length) = \{3,2\}$$

   (b) translates the numerical search as a SMT formula: $5 \geq N \wedge 4 \geq N \wedge \neg(3 \geq N) \wedge \neg(2 \geq N)$
   (c) substitutes numerical variables with a solution to the SMT formula:
   `f(List) ← length(List,Length), leq(Length,3)`

4 - Experiment 1

Q1 Can NUMSYNTH learn programs with numerical values?
Q2 How well does NUMSYNTH perform compared to other approaches?

Task	ALEPH	MAGICPOPPER	NUMSYNTH
interval	69 ± 1	70 ± 0	**99 ± 1**
halfplane	**99 ± 0**	84 ± 7	96 ± 1
zendo1	98 ± 0	68 ± 3	**99 ± 0**
zendo2	51 ± 1	56 ± 1	**96 ± 1**
zendo3	71 ± 1	51 ± 1	**96 ± 1**
zendo4	63 ± 1	52 ± 1	**94 ± 1**
pharma1	82 ± 1	64 ± 3	**99 ± 0**
pharma2	83 ± 1	77 ± 2	**95 ± 1**
pharma3	81 ± 1	82 ± 1	**98 ± 1**
pharma4	76 ± 1	62 ± 2	**92 ± 1**
memberin	49 ± 0	75 ± 4	**97 ± 1**
lastleq	50 ± 0	51 ± 1	**98 ± 1**
nextgeq	50 ± 0	50 ± 0	**92 ± 5**

Table 3: Predictive accuracies

Task	ALEPH	MAGICPOPPER	NUMSYNTH
interval	1 ± 0	**0 ± 0**	**0 ± 0**
halfplane	**1 ± 0**	60 ± 26	2 ± 1
zendo1	25 ± 8	timeout	**10 ± 1**
zendo2	68 ± 18	97 ± 11	**17 ± 1**
zendo3	106 ± 26	112 ± 9	**69 ± 2**
zendo4	147 ± 30	timeout	**76 ± 2**
pharma1	2 ± 0	3 ± 0	**1 ± 0**
pharma2	10 ± 2	7 ± 0	**2 ± 0**
pharma3	24 ± 3	66 ± 5	**20 ± 1**
pharma4	**3 ± 0**	62 ± 2	20 ± 0
memberin	**1 ± 0**	161 ± 38	2 ± 0
lastleq	**0 ± 0**	589 ± 10	13 ± 1
nextgeq	**0 ± 0**	336 ± 17	39 ± 6

Table 4: Learning times

- **NUMSYNTH can outperform existing approaches when learning programs with numerical values.**

5 - Experiment 2

Q3 How well does NUMSYNTH scale with the number of examples?

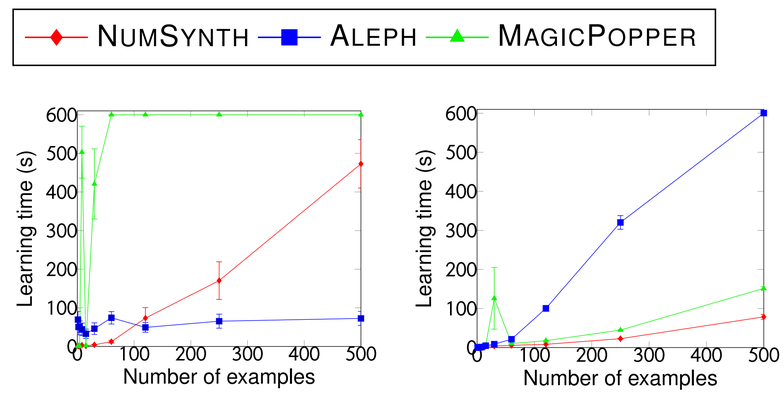


Fig. 2: Learning time versus the number of examples for *zendo1*.

Fig. 3: Learning time versus the number of examples for *pharma2*.

- **NUMSYNTH can scale well, better than ALEPH and MAGICPOPPER with respect to the number of examples**
- **Numerical search stage can be expensive and limit scalability**

6 - Conclusion

- **Efficiently learning programs with numerical values from infinite domains and reason about multiple examples.**
- **NUMSYNTH can outperform existing approaches.**

Future work and Limitations:

- scalability with respect to the complexity of the numerical reasoning stage
- learn numerical values from noisy examples

References

[1] A. Cropper and R. Morel. Learning programs by learning from failures. *Mach. Learn.*, 2021.

[2] R. Evans and E. Grefenstette. Learning explanatory rules from noisy data. *J. of AI Research*, 61:1–64, 2018.

[3] C. Hocquette and A. Cropper. Learning programs with magic values. *Mach. Learn.*, 2022.

[4] S. Muggleton. Inverse Entailment and Progol. *New Generation Comput.*, 13(3&4):245–286, 1995.

[5] A. Srinivasan and R. Camacho. Numerical reasoning with an ILP system capable of lazy evaluation and customised search. *J. Logic Prog.*, 1999.

Article

Code