Reinforcement Learning for SyGuS

Julian Parsert Elizabeth Polgreen

University of Oxford, University of Edinburgh

Syntax-Guided Synthesis (SyGuS)

SyGuS is a problem of synthesising

- a function F within
- a theory τ that satisfies
- a semantic specification ϕ
- with a syntactic restriction G.

$\rightarrow$ SyGuS-IF closely follows SMTLIB

SyGuS Tools/competitions

```
(set-logic LIA)
(synth-fun max2 ((x Int) (y Int)) Int
    ((I Int) (B Bool))
    ((I Int (x y 0 1
             (+ I I) (- I I)
             (ite B I I)))
     (B Bool ((and B B) (or B B) (not B)
              (= I I) (<= I I) (>= I I))))))
(declare-var x Int)
(declare-var y Int)
(constraint (>= (max2 x y) x))
(constraint (>= (max2 x y) y))
(constraint (or (= x (max2 x y)) (= y (max2 x y))))
(check-synth)
```

AI in SyGuS

Many AI based approaches to Synthesis (and SyGuS):

- DeepCoder (Deep Learning for I/O examples)
- Neuro-Symbolic Program Synthesis (Neural Embedding of I/O examples, R3NN syntheses a function from Embedding)
- DreamCoder
- Flash Fill
- ...

$\rightarrow$ All based on I/O or PBE domains

Abstract Domains/Theories with logical specifications?

AI in SyGuS

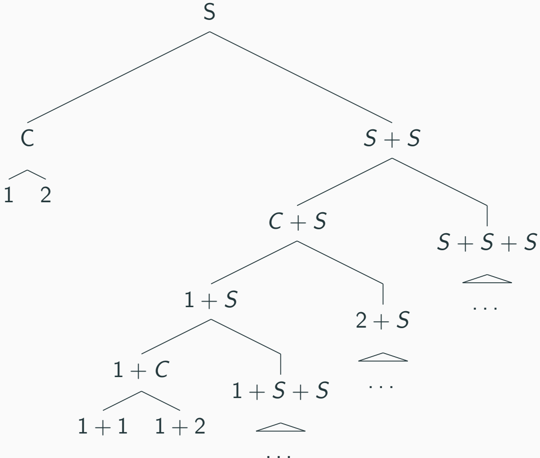
Lack of labeled training data?

Lack of training problems?

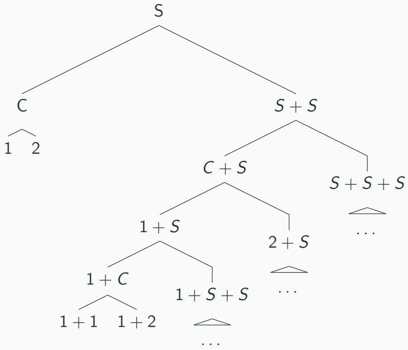
Enumerative Search as a Tree Search

$$S \to S + S \mid C$$

$$C \to 1 \mid 2$$



Game of Synthesis



- *States S* are the vertices
- *Actions A* are the edges
- *Final states* are leaf nodes
- *Winning states* correct final states

Intelligent Tree Search - Policy/Value

Let (E, V) be the Grammar Tree.

Policy

The policy $\pi : E \mapsto \mathbb{R}$ is a function that given an edge $(u, v) \in E$ estimates the likelihood of success when choosing an *action* that leads to v when in state u.

Value

The value $\mathrm{W} : V \mapsto \mathbb{R}$ is a function that given an vertex (i.e. state) $u \in V$ estimates the "quality" of this state.

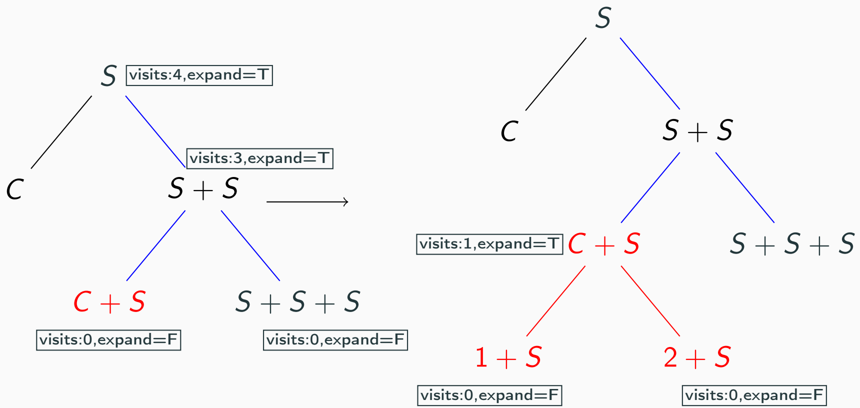
Monte-Carlo Tree Search

Searching game tree in 4 phases:

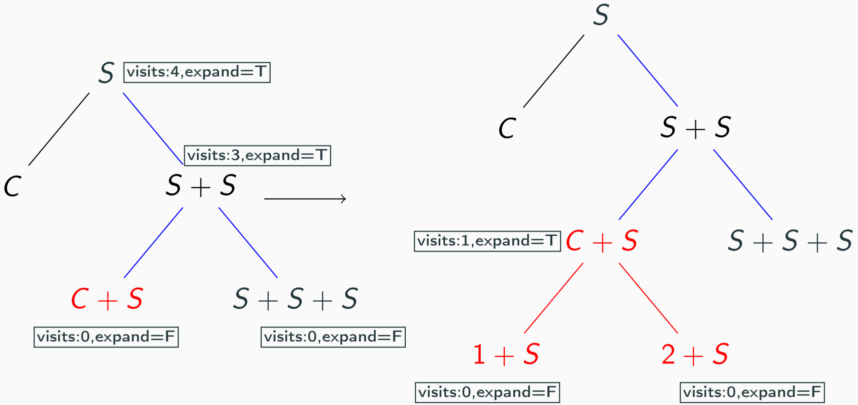
1. Big-step
2. Rollout
3. Expand
4. Backpropagation

During search we keep track of visit count, value, policy in the tree.

Algorithm - Rollout from S with expansion

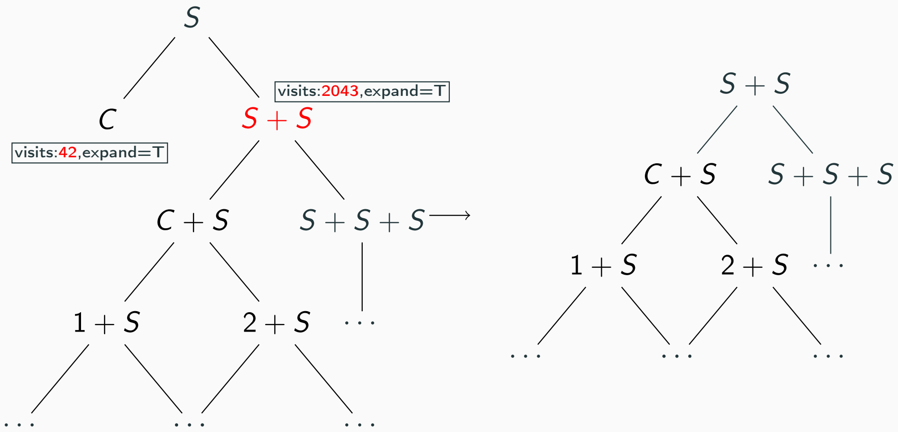


Algorithm - Rollout from S with expansion



$$\mathrm{uct}(p, c) = \frac{\Sigma \mathrm{W}(c)}{\mathrm{N}(c)} + \gamma * \pi(p, c) * \sqrt{\frac{\log \mathrm{N}(p)}{\mathrm{N}(c)}}$$

Algorithm - Decision



Reinforcement Learning

1. Run search on all training Problems with given policy/value functions
2. Collect training data from searches
3. Train policy/value models with new data for next iteration
4. Go to step 1 with new policy/value

Generating SyGuS problems

Generating Training Data

generate (a lot of) SyGuS problems from SMT problems

Given a problem P:

1. get a set S of sub-terms in P.
2. anti-unification on S, least general generalization (LGG) L
3. replace terms S in P with variable F
4. use unification for arguments for F
5. translate P (with F) to SyGuS problem P_s.

```
(set-logic LIA)
(assert 10 * x = (2 * x) + y)
(assert x * 3 + 5 = 8)
(check-sat)
```

sat $x = 1, y = 8$

$$10 * 1 = (2 * 1) + 8$$
$$(1 * 3) + 5 = 8$$

1. Choose a set of Terms

$$10 * 1 = (2 * 1) + 8$$

$$(1 * 3) + 5 = 8$$

1. Choose a set of Terms

$$10 * 1 = (2 * 1) + 8$$

$$(1 * 3) + 5 = 8$$

1. Choose a set of Terms

$$10 * 1 = (2 * 1) + 8$$

$$(1 * 3) + 5 = 8$$

2. Anti-Unification on Terms

$$(2 * 1) + 8 \sqcup (1 * 3) + 5$$

1. Choose a set of Terms

$$10 * 1 = (2 * 1) + 8$$

$$(1 * 3) + 5 = 8$$

2. Anti-Unification on Terms

$$(2 * 1) + 8 \sqcup (1 * 3) + 5$$

$$\rightsquigarrow (2 * 1) \sqcup (1 * 3) + 8 \sqcup 5$$

1. Choose a set of Terms

$$10 * 1 = (2 * 1) + 8$$

$$(1 * 3) + 5 = 8$$

2. Anti-Unification on Terms

$$
\begin{aligned}
&(2 * 1) + 8 \sqcup (1 * 3) + 5 \\
&\rightsquigarrow (2 * 1) \sqcup (1 * 3) + 8 \sqcup 5 \\
&\rightsquigarrow (2 \sqcup 1) * (1 \sqcup 3) + w
\end{aligned}
$$

1. Choose a set of Terms

$$10 * 1 = (2 * 1) + 8$$
$$(1 * 3) + 5 = 8$$

2. Anti-Unification on Terms

$$\begin{aligned} &(2 * 1) + 8 \sqcup (1 * 3) + 5 \\ &\rightsquigarrow (2 * 1) \sqcup (1 * 3) + 8 \sqcup 5 \\ &\rightsquigarrow (2 \sqcup 1) * (1 \sqcup 3) + w \\ &\rightsquigarrow u * v + w \end{aligned}$$

lgg $u * v + w$ with 3 new variables

3. Replace terms with $F(u, v, w)$

$$10 * 1 = (2 * 1) + 8$$

$$(1 * 3) + 5 = 8$$

3. Replace terms with $F(u, v, w)$

$$10 * 1 = F(u, v, w)$$

$$F(u, v, w) = 8$$

3. Replace terms with $F(u, v, w)$

$$10 * 1 = F(u, v, w)$$

$$F(u, v, w) = 8$$

4. Unification of $u * v + w$ with terms for arguments

$$(2 * 1) + 8 \stackrel{?}{=} u * v + w \qquad u \mapsto 2, v \mapsto 1, w \mapsto 8$$

$$(1 * 3) + 5 \stackrel{?}{=} u * v + w \qquad u \mapsto 1, v \mapsto 3, w \mapsto 5$$

$$10 * 1 = F(2, 1, 8)$$

$$F(1, 3, 8) = 8$$

```
(set-logic LIA)
(synth-fun F ((u Int) (v Int) (w Int)) Int
  ((I Int))
  ((I Int (u v w 0 1
      (+ I I) (∗ I I)))
  ))
(constraint (= (10 ∗ 1) (F 2 1 8)))
(constraint (= (F 1 3 8) 8))
(check-synth)
```

```
(set-logic LIA)
(assert 10 ∗ x = (2 ∗ x) + y)
(assert x ∗ 3 + 5 = 8)
(check-sat)
```

EXPERIMENTAL EVALUATION

Implementation

- Previous Iteration in Python
- Implementation in C++
- Verification with CEGIS loop and Z3
- size 2 term walks as features
- Gradient Boosted Trees (xgboost) as ML models

Research Questions

How well does this work?

How does it compare to other tools?

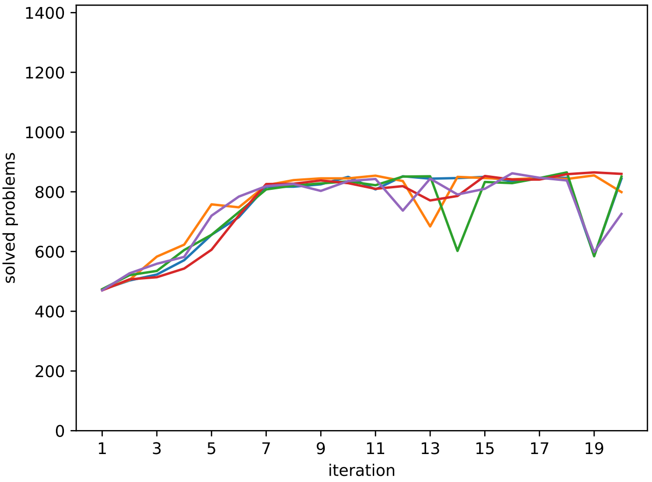
Experimental Evaluation

- 1425 SyGuS *training* problems and 476 *testing* problems
- 100 second timeout

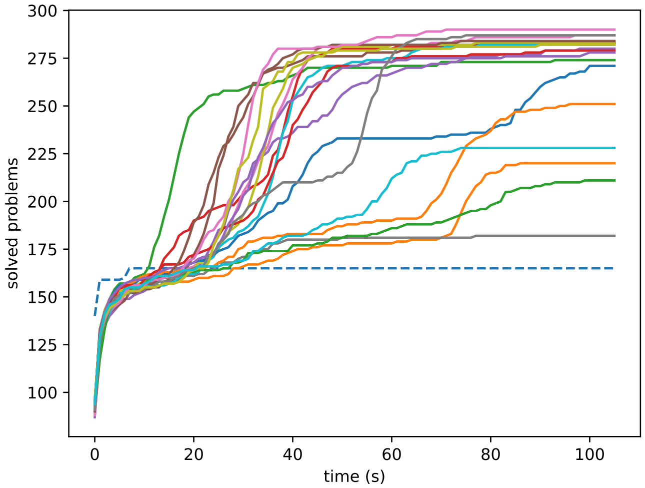
471.6 and 163.4 solved in first iteration

set	total	min	max	mean	stdev
train	1425	852	865	859.8	5.97
test	476	287	292	289	1.87

Problems solved in each iteration



Problems solved over time in each iteration



Compared to State of the Art?

CVC5 solves 834 on training and 295 on Testing set

set	total	min	max	mean	stdev
train	1425	852	865	859.8	5.97
test	476	287	292	289	1.87

Conclusion

Summary

- Enumerative synthesis as tree search
- AlphaZero style Monte-Carlo based tree search
- Learned policy/value functions with UCT for guidance
- Reinforcement learning for policy/value models
- SyGuS problem Generation
- Implementation based on xgboost trained and tested entire set

Preprint: https://arxiv.org/abs/2307.09564

Questions?