Neural-network guided Grammar Filtering For Syntax-Guided Synthesis

Kairo Morton†, William T. Hallahan*, Elven Shum*, Ruzica Piskac*, Mark Santolucito††

*Yale University. †MIT, ††Barnard College, Columbia University
Solvers are Supplemented with Additional Techniques

• For CEGQL:
  • Partial quantifier elimination as a preprocessing pass
  • Heuristic solution reconstruction

• For enumerative:
  • Divide-and-conquer [Alur et al 2017]
  • Piecewise-Independent Unification (UNIF+PI) [Barbosa et al FMCAD2019]
  • Theory-specific constant repair [Abate et al 2019]
  • Static grammar minimization and symmetry breaking
  • Variable agnostic enumeration
Solvers are Supplemented with Additional Techniques

• For CEGQI:
  • Partial quantifier elimination as a preprocessing pass
  • Heuristic solution reconstruction

• For enumerative:
  • Divide-and-conquer [Alur et al 2017]
  • Piecewise-Independent Unification (UNIF+PI) [Barbosa et al FMCAD2019]
  • Theory specific constant repair [Abate et al 2019]
  • Static grammar minimization and symmetry breaking
  • Variable agnostic enumeration
Standard SyGuS synthesis flow
Standard SyGuS synthesis flow

- Specification
  - Constraints
  - Grammar

- SyGuS Solver

- Code
Standard SyGuS synthesis flow

```
(synth-fun f ((x String)) String
 ((Start String (ntString)))
 (ntString String
  (x
   (str.++ ntString ntString)
   (str.at ntString ntInt) )))
 (ntlnt Int
  (0 1 2 3 4 5
   ( + ntlnt ntlnt) (- ntlnt ntlnt)
   (str.len ntString)
   ))

(declare-var x String)
(constraint (= (f "Hello") "HelloHello"))
```
Standard SyGuS synthesis flow

\begin{itemize}
  \item \texttt{(synth-fun f ((x String)) String (Start String (ntString))) (ntString String (x (str.++ ntString ntString) (str.at ntString ntInt))) (ntInt Int (0 1 2 3 4 5 (+ ntInt ntInt) (- ntInt ntInt) (str.len ntString)))}
  \item \texttt{(declare-var x String)}
  \item \texttt{(constraint (= (f "Hello") "HelloHello"))}
\end{itemize}

\begin{itemize}
  \item \texttt{(define-fun f ((x String)) String (str.++ x x))}
\end{itemize}
Standard SyGuS synthesis flow

```lisp
(synth-fun f ((x String)) String
  ((Start String (ntString)))
  (ntString String
    (x
     (str.++ ntString ntString)
     (str.at ntString ntInt)))
  (ntInt Int
    (0 1 2 3 4 5
     (+ ntInt ntInt) (- ntInt ntInt)
     (str.len ntString)))
)

(declare-var x String)
(constraint (= (f "Hello") "HelloHello"))

(define-fun f ((x String)) String (str.++ x x))
```
Predicting the grammar to reduce the search space

\[
\begin{align*}
\text{(synth-fun } f \ ((x \ \text{String})) & \text{ String} \\
& ((\text{Start String} \ (nt\text{String}))) \\
& (nt\text{String} \ \text{String} \\
& \ (x \\
& \ (\text{str.++ } nt\text{String} nt\text{String}) \\
& \ (\text{str.at } nt\text{String} ntlnt)) \\
& (nt\text{Int} \ \text{Int} \\
& \ (0 \ 1 \ 2 \ 3 \ 4 \ 5 \\
& \ (+ ntlnt ntlnt) (- ntlnt ntlnt) \\
& \ (\text{str.len } nt\text{String}) \\
& ))
\end{align*}
\]

\[
\begin{align*}
\text{(declare-var } x \ \text{String}) \\
\text{(constraint } (= (f "Hello") "HelloHello"))
\end{align*}
\]

\[
\begin{align*}
\text{(define-fun } f \ ((x \ \text{String})) & \text{ String} (\text{str.++ } x \ x))
\end{align*}
\]
Components of grammar prediction

- Constraints
- Grammar
- Learn Semantics
- Learn Reasoning
- Predict Time
- Criticality
- Combine
- Filtered Grammar
- SyGuS Solver
- Code
Component Models

Criticality : (IOExample, Terminal) → [0,1]

Predict Time : Terminal → TimeSaved (ℝ)

Combine : [[0,1], TimeSaved ] |G| → Boolean
**Component Models**

**Criticality**: \((\text{IOExample}, \text{Terminal}) \rightarrow [0,1]\)

What is the probability a terminal is critical for this constraint?

**Predict Time**: \(\text{Terminal} \rightarrow \text{TimeSaved} (\mathbb{R})\)

**Combine**: \([0,1], \text{TimeSaved} \rightarrow \text{Boolean}\)
Component Models

**Criticality** : (IOExample, Terminal) → [0,1]
What is the probability a terminal is critical for this constraint?

**Predict Time** : Terminal → TimeSaved (ℝ)
What is the estimated time saved by removing this terminal?

**Combine** : [[0,1], TimeSaved ]^{|G|} → Boolean
**Component Models**

**Criticality** : (IOExample, Terminal) → [0,1] ⌣G
What is the probability a terminal is critical for this constraint?

**Predict Time** : Terminal → TimeSaved (ℝ)
What is the estimated time saved by removing this terminal?

**Combine** : [[0,1], TimeSaved ] ⌣G → Boolean
Of the terminals that save the most time by being removed, which are least likely to be critical?
Obtaining training data for Criticality

Criticality : (IOExample, Terminal) → [0,1]
Obtaining training data for Criticality

**Criticality** : (IOExample, Terminal) → [0,1]

1) Generate functions \( f \) from the grammar (`--sygus-stream` with no constraints)

```
(define-fun f ((x String)) String x)
(define-fun f ((x String)) String (str.++ x x))
(define-fun f ((x String)) String (str.++ x (str.++ x x)))
[...]
```
Obtaining training data for Criticality

1) Generate functions \( f \) from the grammar (--sygus-stream with no constraints)
2) Apply each function \( f : \text{String} \rightarrow \text{String} \) to randomly generated inputs

\[
\begin{align*}
(\text{define-fun } f \ (\ (x \ \text{String}) \ ) \ \text{String} \ x) & \quad \text{f(x) = x ++ x ++ x} \\
(\text{define-fun } f \ (\ (x \ \text{String}) \ ) \ \text{String} \ (\text{str.++ x x})) & \quad \text{f("rand") = "randrandrand"} \\
(\text{define-fun } f \ (\ (x \ \text{String}) \ ) \ \text{String} \ (\text{str.++ x (str.++ x x)})) & \quad \text{f("All") = "AllAllAllAll"} \\
[...] & \quad \text{...}
\end{align*}
\]
Obtaining training data for Criticality

1) Generate functions $f$ from the grammar (--sygus-stream with no constraints)
2) Apply each function $f : \text{String} \rightarrow \text{String}$ to randomly generated inputs
3) Build “one-hot vector” of IOExamples and terminals used in $f$

(define-fun f ((x String)) String x)  \quad f(x) = x ++ x ++ x
(define-fun f ((x String)) String (str.++ x x))  \quad f(“rand”) = “randrandrand”
(define-fun f ((x String)) String (str.++ x (str.++ x x)))  \quad f(“AllI”) = “AllIAllIAllIAllI”
[...]

Criticality : (IOExample, Terminal) → [0,1]
SyGuS-Comp is a yearly SyGuS solver competition

PBE-Strings track has 64 programming by example problems over strings, integers and Booleans

Solvers have an hour to solve each problem

In 2019, CVC4 won the PBE-Strings track

Q: Can GRT + CVC4 outperform CVC4 on the PBE-Strings competition benchmarks?
Evaluation

47.6% faster

When timing data is not considered, and we consider only criticality, synthesis succeeds on only 11 out of the 64 benchmarks.

On **32 out of 64** benchmarks **GRT + CVC4 is faster** than just CVC4. On 31 benchmarks, performance is the same (within ± .1 second.) In total, **CVC4 solves 62** of the benchmarks, **GRT + CVC4 solves 63**. The extra benchmark is solved in 3516.21 seconds.
Future directions

Taking not just criticality, but also potential time savings into account is needed to effectively filter SyGuS grammars.

Open questions:

- Would a more sophisticated time prediction heuristic be more effective?
- Could all examples be encoded and given to the NN together, to allow learning about interaction between constraints?
- How could this be adapted to work with other common input types (including integers, bitvectors, and algebraic datatypes)?
- How could general first-order logic constraints be encoded in the NN?
Kairo Morton, William T. Hallahan, Elven Shum, Ruzica Piskac, Mark Santolucito
Grammar Filtering for Syntax-Guided Synthesis. AAAI 2020

https://github.com/KTMorton/Live_Programming_Research
Kairo Morton, William T. Hallahan, Elven Shum, Ruzica Piskac, Mark Santolucito
*Grammar Filtering for Syntax-Guided Synthesis*. AAAI 2020

[https://github.com/KTMorton/Live_Programming_Research](https://github.com/KTMorton/Live_Programming_Research)

Nicolas Chan, Elizabeth Polgreen, Sanjit A. Seshia