Syntax Error Correction

ANONYMOUS AUTHOR(S)

In this work, we illustrate how to lower context-free language recognition onto a tensor algebra over finite fields. In addition to its theoretical value, this connection has yielded surprisingly useful applications in incremental parsing, code completion and program repair. For example, we use it to repair syntax errors, perform sketch-based program synthesis, and decide various language induction and membership queries. This line of research provides an elegant unification of context-free program repair, code completion and sketch-based program synthesis. To accelerate code completion, we design and implement a novel incremental parser-synthesizer that transforms CFGs onto a dynamical system over finite field arithmetic, enabling us to suggest syntax repairs in-between keystrokes.

The tool accepts a sequence of lexical tokens and suggests a set of possible syntax repairs.

For example, consider the following text:

```
 Original:  val splitTokens : MutableList < String > = ArrayList ( )
Corrupted:  var splitTokens : MutableList < String > = ArrayList ( )
-------------------------------------------------------------------------
D=1 repair: var splitTokens : MutableList < String > = ArrayList ( )
D=1 repair: val splitTokens : MutableList < String > = ArrayList ( )
D=2 repair: var protected : MutableList < String > = ArrayList ( )
D=2 repair: var splitTokens : MutableList < tailrec > = ArrayList ( )
D=2 repair: var splitTokens : MutableList < String > = inline ( )
D=2 repair: val splitTokens : MutableList < override > = ArrayList ( )

Found 426 valid repairs in 13062ms, or roughly ~32.61 repairs per second.
Original string was #1 in repair proposals!
```

SPLASH'23, October 22-27, 2023, Cascais, Portugal
2023. ACM ISBN 978-x-xxxx-xxxx-x/YY/MM…$15.00
https://doi.org/10.1145/nnnnnnn.nnnnnnn