SYSERTSKITE:
Syntactic & Semantic guidance from Refinement Types for Synthesis, via a toKen-based Incremental Type-checking Engine

Stéphane Graham-Lengrand, SRI International

https://www.csl.sri.com/users/sgl/sysertskite

Challenge: LLMs alone cannot be trusted to synthesize correct code.

Synthesizing correct code / checking output code correctness may require performing arbitrarily complex computation (e.g., compute a hash) or reasoning (e.g., prove a theorem).

LLMs alone cannot be trusted to do that correctly.

Solution: Involve external trustworthy tools (e.g., code interpreter, formal verification tool)

Possible approaches:

1. Train LLM to select appropriate tools + produce the inputs they will run on.

   Can it be trusted to use external tools correctly?

2. Use trustworthy tools to verify LLM's complete output
   + iterate calls to LLM with new prompts until the output is correct.

   A posteriori checking means search space remains huge at every iteration. Convergence?

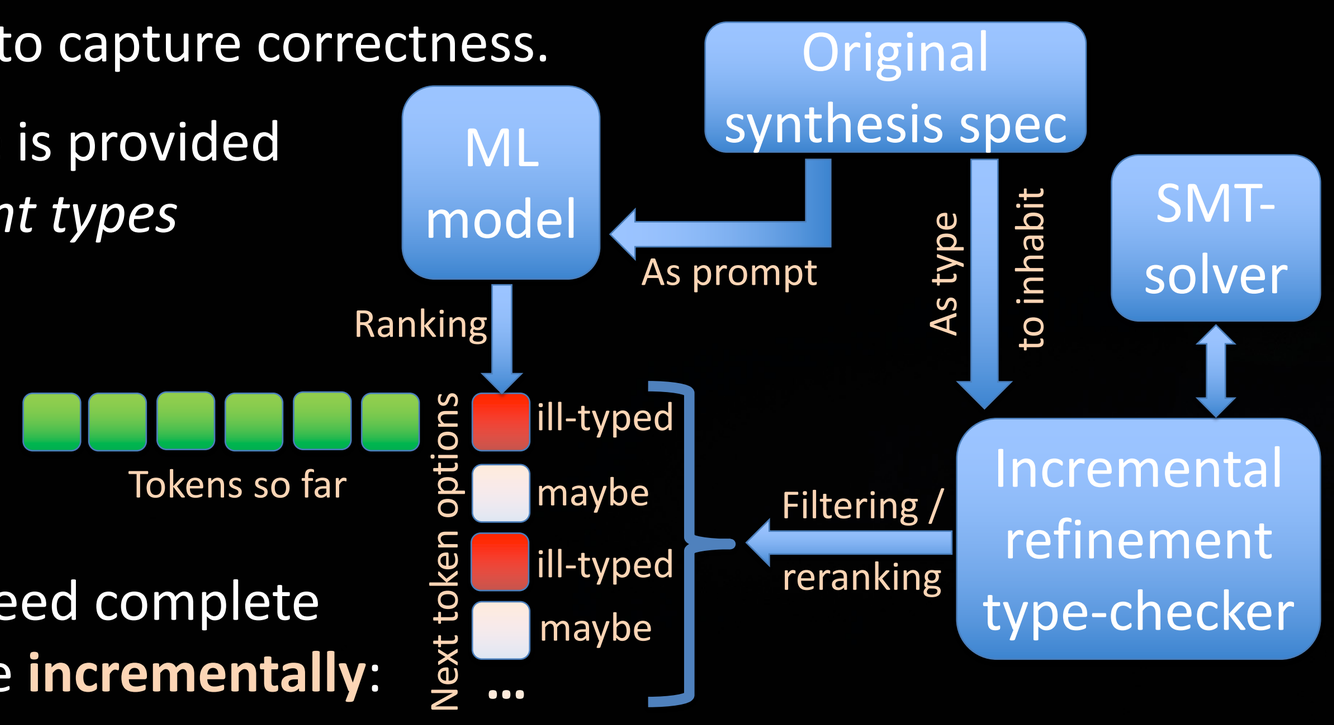
3. Use trustworthy tools to **dynamically guide LLM output**, token-per-token, towards **correct-by-construction output**. Guidance means search space is **reduced** by external tool.

Hope: Single prompt is enough & model size can be a lot smaller.

Classic case in state of the art: force output to adhere to grammar (*Syntax-Guided Synthesis*).

But adhering to grammar is rarely enough to capture correctness.

SYSERTSKITE: Approach 3. where guidance is provided by syntactic + semantic specs via *refinement types* (a.k.a. *predicate subtypes*) + SMT-solving.



- Much **richer than syntactic guidance**; correctness can often be captured with semantics specs;
- **Novel type-checking engine** does not need complete output; it checks output token sequence **incrementally**:
- At each step, it **detects and filters out ill-typed tokens** that would never lead to correct code; tokens can be reranked if type-checking can symbolically infer good tokens ahead;
- Naturally fits *beam search* technique; can be combined with approach 2. when reaching dead-ends; can be used in a *Monte-Carlo Tree Search* instead of LLM sequence production.

Example synthesis task: Synthesize sequence of instructions (i.e., sequence of tokens) to transform state (x = 0, count = 0) to state where count = 1000. Only two kinds of instructions:

`evenHash()` mutates (x, count) to (hash(x), count+1) but crashes if x is odd

`oddHash()` mutates (x, count) to (hash(x), count+1) but crashes if x is even

- Only one single correct solution. Finding the correct sequence of 1000 instructions requires computing hash precisely, otherwise 2^{-1000} probability of getting it right. LLMs alone fail.
- Approach 1 works by "using code interpreter", but fails at minor variant that requires symbolic reasoning. SYSERTSKITE drives the synthesis of correct sequence by eliminating crashing tokens.

C3E
Computational Cybersecurity in Compromised Environments
2024 Fall Workshop | September 17-19 | SRI International in Menlo Park, CA