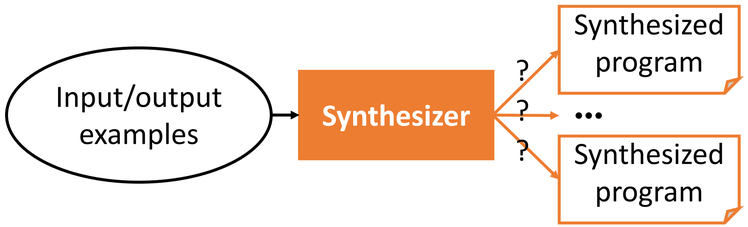
Using Active Learning to Synthesize Models of Applications that Access Databases

Jiasi Shen (jiasi@csail.mit.edu), Martin Rinard

Massachusetts Institute of Technology

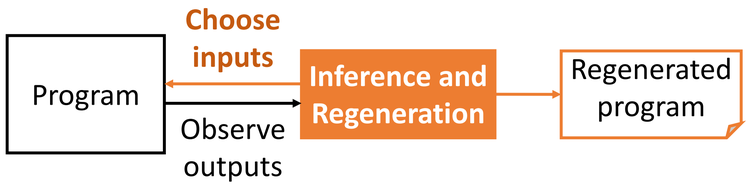
Background: Inductive program synthesis



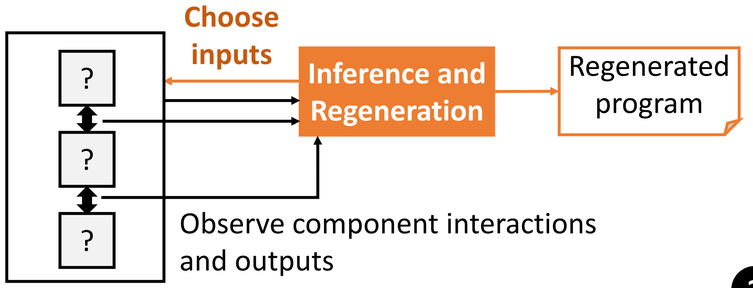
Often underspecify program behavior
Not necessarily easier to write than the program

1

Program inference and regeneration

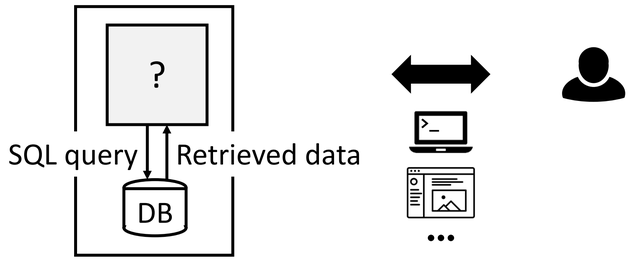


Leverage a program as the specification
Use **active learning,** eliminate uncertainty
Migrate, reverse engineer, rewrite legacy code



2

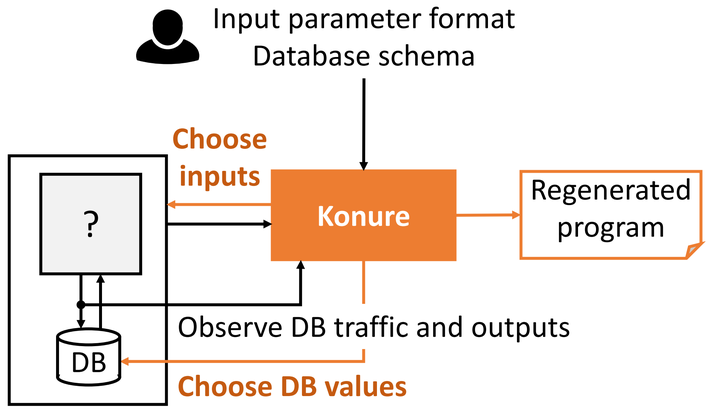
Data-retrieval applications



Prevalent, complex imp., simple core functionality
Data flow often manifests as SQL queries
Control flow largely depends on query results

3

Component-based program inference



4

Konure DSL and inference algorithm

$$
\begin{array}{lcl}
\textbf{Prog} & := & \epsilon \mid \textbf{Seq} \mid \textbf{If} \mid \textbf{For} \\
\textbf{Seq} & := & \textbf{Query Prog} \\
\textbf{If} & := & \texttt{if } \textbf{Query} \texttt{ then } \textbf{Prog} \texttt{ else } \textbf{Prog} \\
\textbf{For} & := & \texttt{for } \textbf{Query} \texttt{ do } \textbf{Prog} \texttt{ else } \textbf{Prog}
\end{array}
$$

Use DSL to represent inferrable program functionality
Use DSL sentential form to represent hypothesis
Use active learning to generate inputs
Resolve subprograms recursively, no backtracking

5

Soundness and completeness

Given any program in DSL,
Konure infers a correct program.

PLDI 2019, TOPLAS (to appear)

6

Experimental results

Applied to several Java and Ruby on Rails applications
Synthesized new implementations in Python

http://bit.ly/konure

7