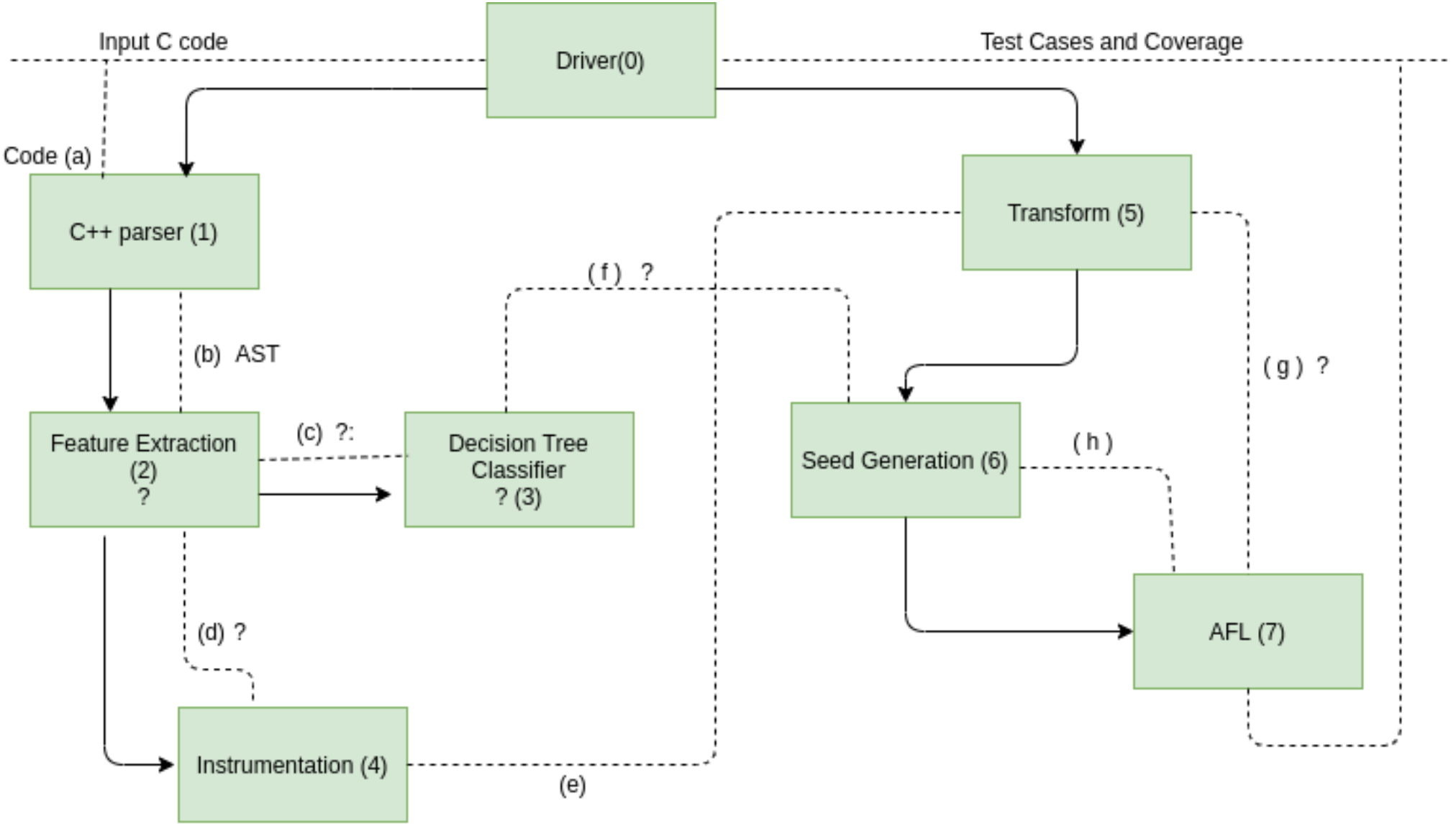
Input C code
Driver(0)
Test Cases and Coverage
Code (a)
C++ parser (1)
Transform (5)
(f) ?
(b) AST
(g) ?
Feature Extraction (2) ?
(c) ?:
Decision Tree Classifier ? (3)
Seed Generation (6)
(h)
AFL (7)
(d) ?
Instrumentation (4)
(e)

Naming Convention

Dotted Line – Input / Output to each box

Solid Line – Execution Sequence

Numbers (1..n) – Execution order

Alphabets (a..z) – Input/Output Labels

Verifuzz.py --> (contains (0) from diagram)

- process cmd line arg
- Set enviornment variables
- Make temp directories
- Go to preProcessInputFile.py via command
- Goes to runVerifuzz.py

preProcessingInputFile.py--->(contains (1,2,4) from diagram)

- Process cmd arg and set env vars
- Creates a .prj file and write some data
- Calls a cppfe command for IR generation
- Runs cmmand to invote fuzzManager.RunFuzzingAnalysis java object file for performing feature extraction
- Uses above output and runs "classifyFuzzVariant.py"
- Run the same fuzzManager.RunFuzzingAnalysis java object file with different input and output directories for performing instrumentation

ClassifyFuzzVariant.py ---->(contains (3) from diagram)

- Loads and classify the program and dumps result in algoCategory.txt using decision tree classifier

runVerifuzz.py -->

- Sets appropiate env vars
- Runs exeFuzzer.py with different AFL, Flags AlgoCategory, Property (Branch Coverage ,Branch error,etc)

exeFuzzer_veriFuzz.py---> (contains (5,6,7) from diagram)

- Determines type of algo using AlgoCategory.txt
- Transforms the instrumented c code with “transform” binary
- Compiles the transformed code
- Sets some Afl Flags According to Requirement
- randomSeedGenCompile with either vs32.o vs64.o byteTrackerForRangeAnal.c for Random seed generation dumped in file name s1_default
- If failed then try whitbox seedGen
- And then much code about generating seed with different techniques and generating witness files
- Finally run AFL with Above output