

Timely Feedback on Assembly Assignments using KLEE

Tingkai Liu

with Zikai Liu, Qi Li, Wenqing Luo, and Steven S. Lumetta



ZJU-UIUC Institute

Zhejiang University / University of Illinois at Urbana-Champaign Institute



Motivation: Students Need Immediate Feedback



- ECE220 at ZJU-UIUC Institute:
Fall 2018: 1 Professor, 2 TAs, 79 students.
Fall 2020: 1 Professor, 4 TAs, 109 students.
- At any time (24/7), some student is working.
- Staff work hard, but not available 24/7.
- Basic idea: use computers to give students fast, focused feedback.
- OUR GOAL: Failure cases specific to student code within 5 minutes, available 24/7.

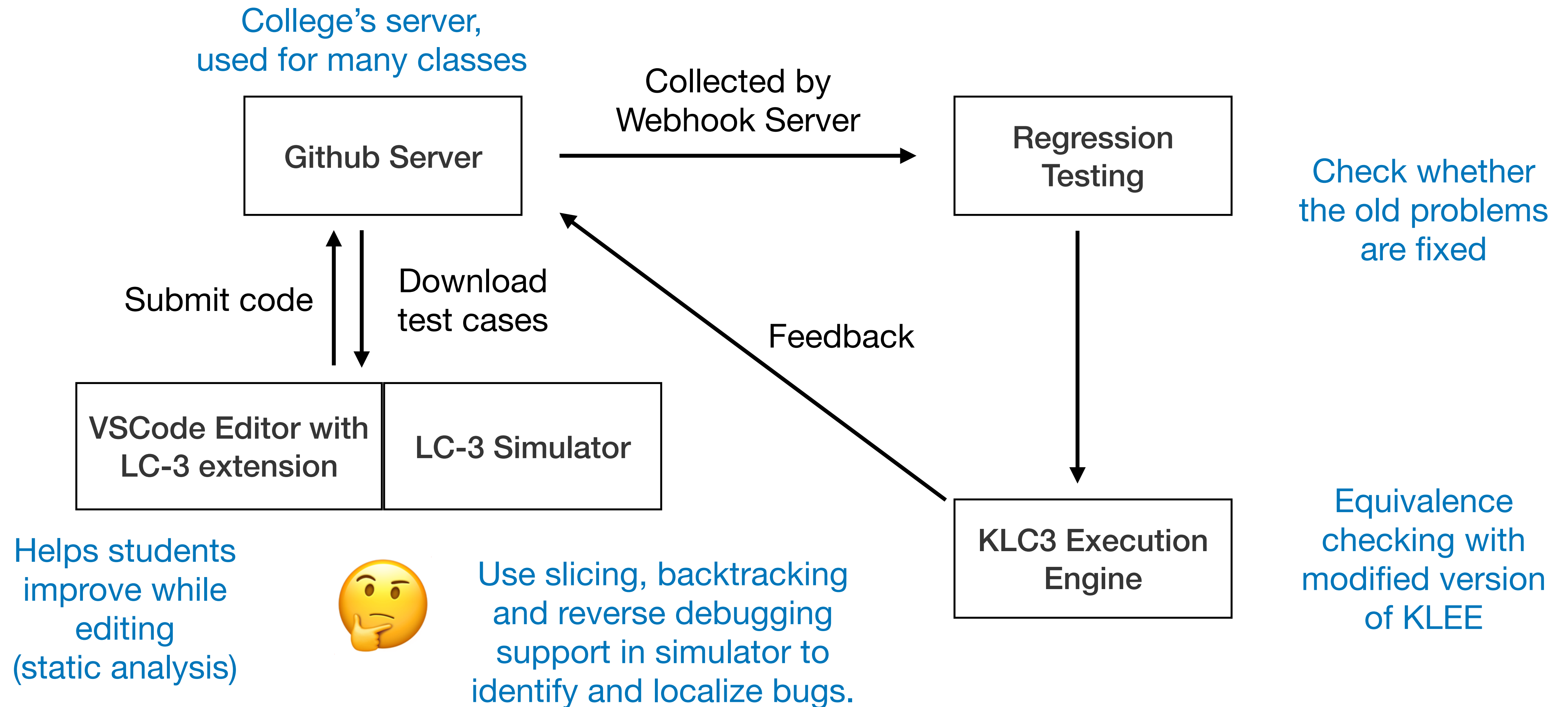


Symbolic Execution is Good at Finding Mistakes

```
int abs (int num) {  
    if (num >= 2) return num;  
    else return -num;  
}
```

	Fixed Test Cases		Symbolic	
Input	42	-220	x	x
Output	42	220	$x (x \geq 2)$	$-x (x < 2)$
Correct?	✓	✓	✓	✗ When $x = 1$ for example

Solution: Developed Tools for Feedback on LC-3 Assembly



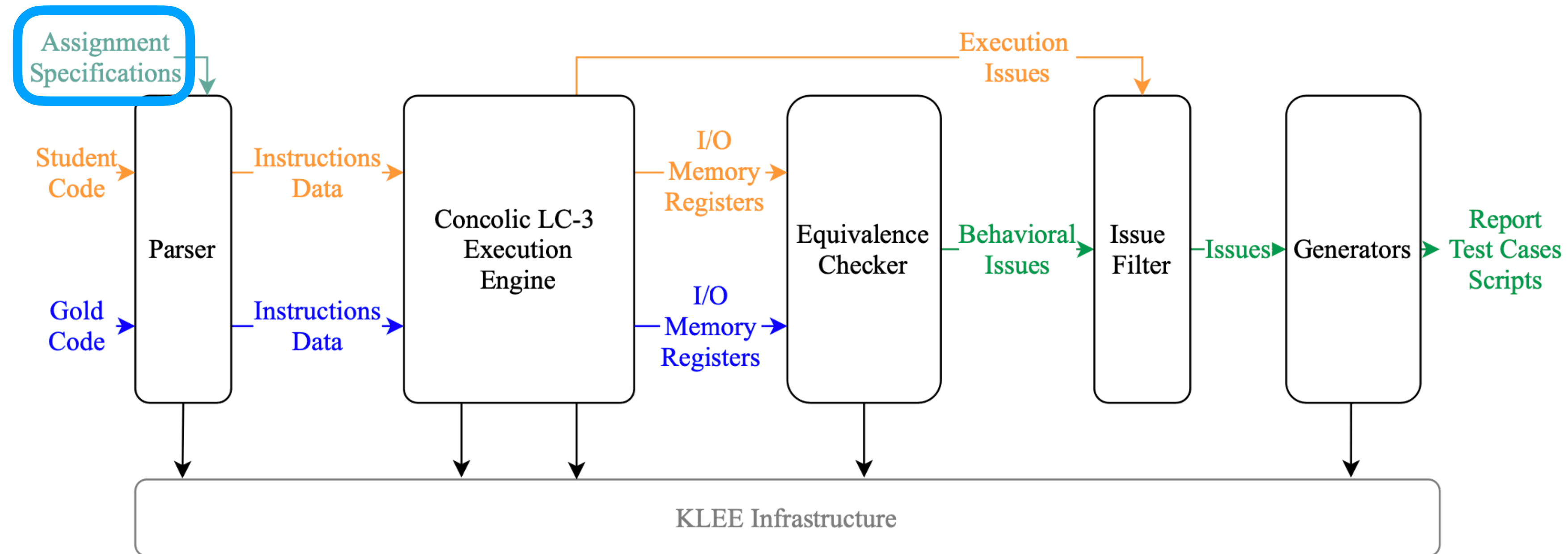
The VSCode Extension: Feedback while Programming LC-3

- Visual Studio Code (VSCode) editor is chosen by many senior students.
- Highlights syntax
- Performs Static Analysis
- Identifies common issues

```
1  .ORIG x3000
2
3  ; TEST: Immediate value
4  ADD R0, R0, #32
5  AND R1, R1, #16
6  ADD R2, R2, x1F
7
8  .END
9
```

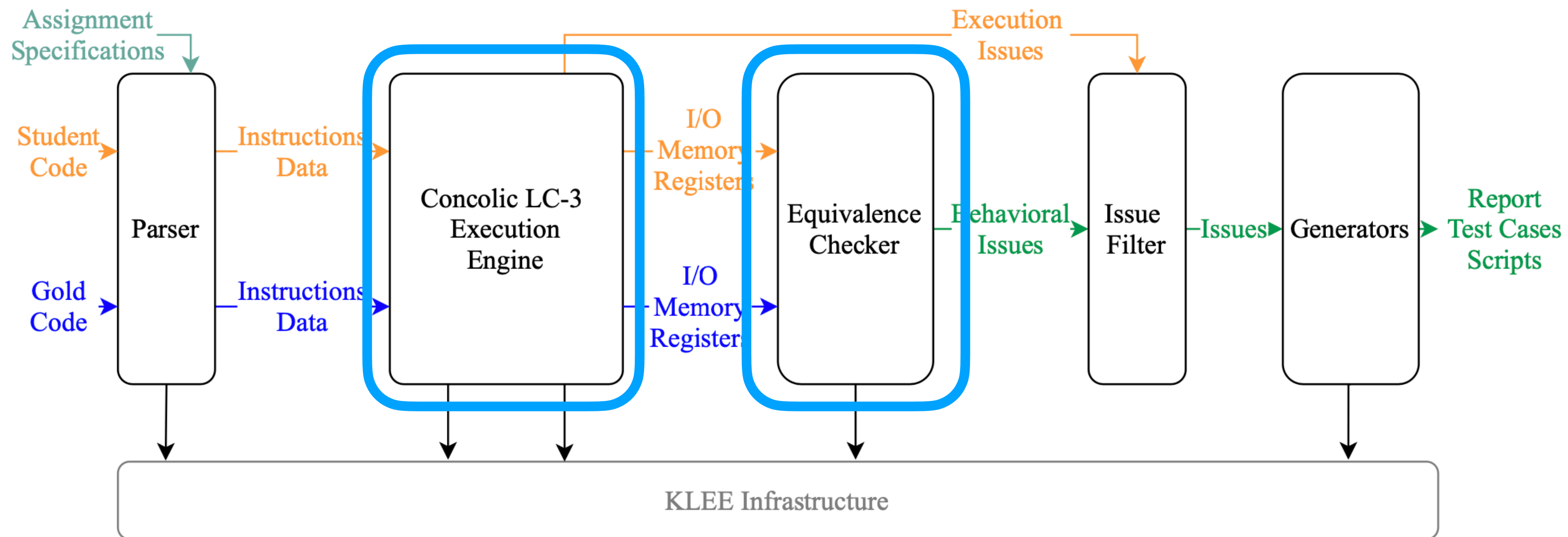
Reduces pressure on the symbolic execution engine.

LC-3 Assembly Extension to Define Input Space



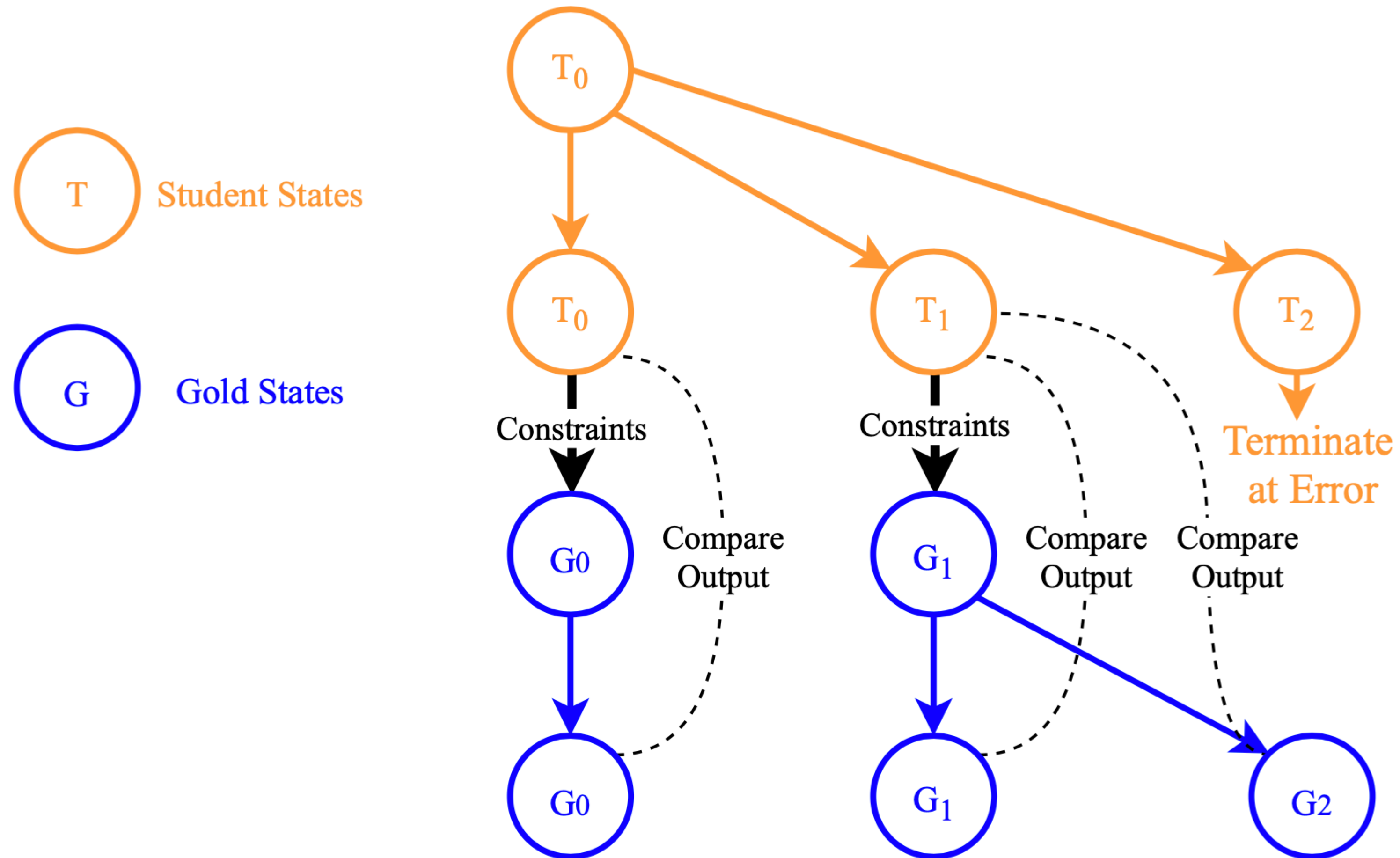
```
.ORIG x4000
.BLKW #1 ; KLC3: SYMBOLIC as INPUT_NUM
        ; KLC3: SYMBOLIC INPUT_NUM >= #0
        ; KLC3: SYMBOLIC INPUT_NUM < #3
.BLKW #8 ; KLC3: SYMBOLIC as INPUT_STR[8]
        ; KLC3: SYMBOLIC INPUT_STR is fixed-length-string
.END
```

LC-3 Assembly Code Executor

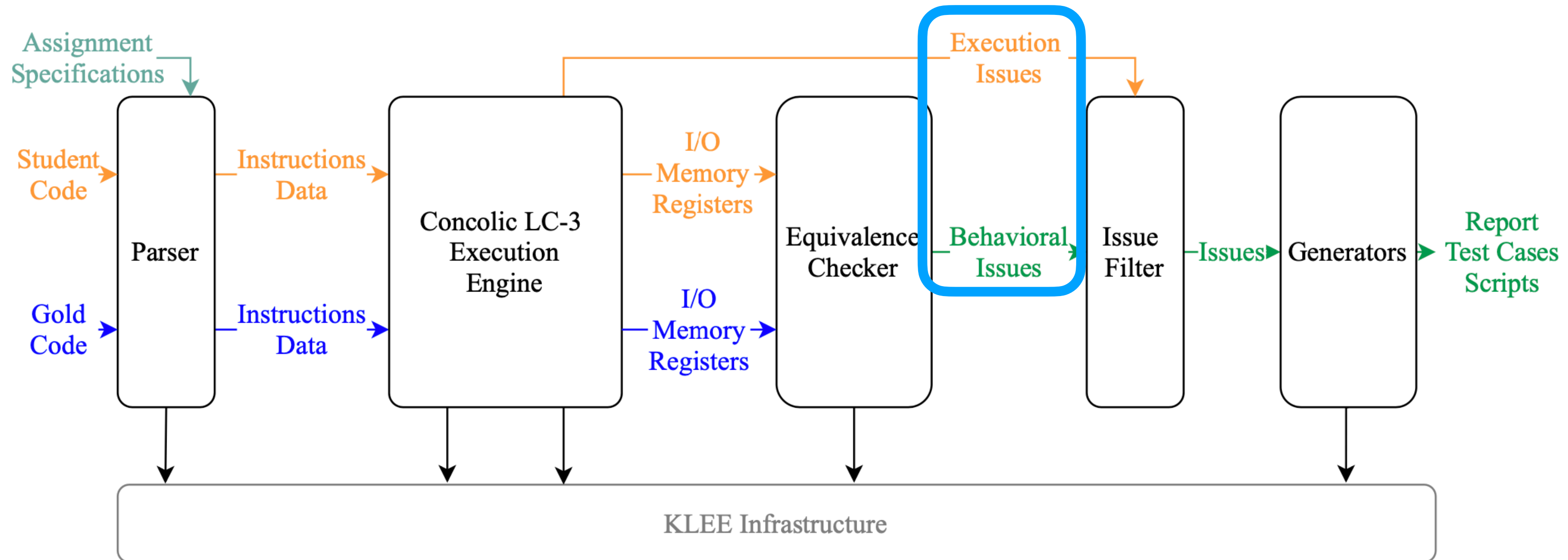


- LC-3 executor and memory model
- Identifies execution issues in student code (orange path)

How Equivalence Checking is Done



Issues are Filtered to Avoid Overwhelming Students



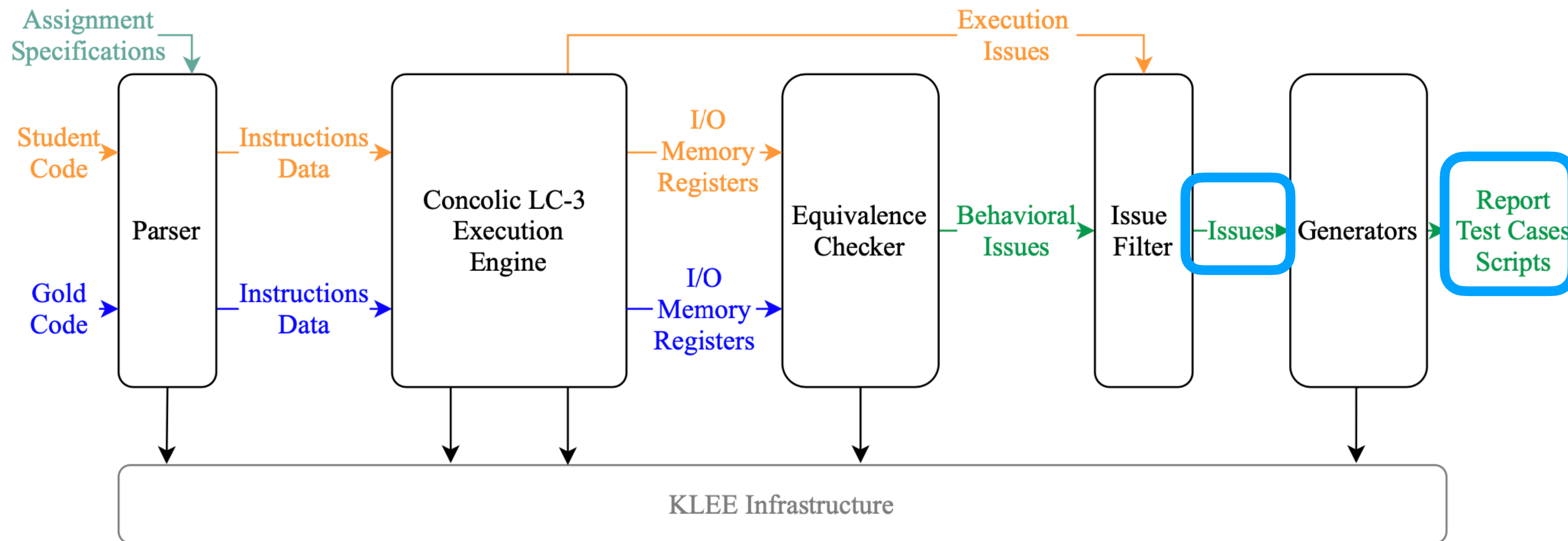
Execution Issues

INPUT_NUM = 0	Line 6	Use uninitialized register R0
INPUT_NUM = 0	Line 12	Access unspecified memory x5000
INPUT_NUM = 1	Line 12	Access unspecified memory x5001
INPUT_NUM = 2	Line 12	Access unspecified memory x5002

Behavioral Issues

INPUT_NUM = 2		Incorrect output
---------------	--	------------------

Issues are Filtered to Avoid Overwhelming Students



Issue Type	Input	Line	Description	Filtering Criteria
Execution Issues	INPUT_NUM = 0	Line 6	Use uninitialized register R0	Same type Same location
	INPUT_NUM = 0	Line 12	Access unspecified memory x5000	
	INPUT_NUM = 1	Line 12	Access unspecified memory x5001	
	INPUT_NUM = 2	Line 12	Access unspecified memory x5002	
Behavioral Issues	INPUT_NUM = 2		Incorrect output	

Reports, Test Cases and Replay Scripts

 **ERROR:** incorrect output.

NOTE: in [test0](#), at step 443, your output (length = 37) is:

```
INCORRECT
```

The expected output (length = 35) is:


```
AAAAAAA
```

Spaces may not be shown clearly. Selecting the text may help.

 **WARNING:** use uninitialized register at [[test.asm:6](#)] `ADD R0, R0, R1`.

NOTE: in [test1](#), at step 3, using uninitialized R0.

REMARK: Do not assume the initial value of register. Now it is assumed to be 0 (notice: not necessarily when replay)

 **WARNING:** read a memory address outside the expected regions at [[test.asm:11](#)] `LDR R3, R0, #0`.

NOTE: in [test1](#), at step 5, reading addr x5000.

REMARK: Remember that uninitialized memory locations have undefined value. Make sure you have calculated address correctly. If it is the location you want to read, make sure you have initialized it.

test0-input.asm

```
; Test case generated by KLC3
.ORIG x4000
.FILL x0002           ; INPUT_NUM
.STRINGZ "AAAAAAA"   ; INPUT_STR
.END
```

test1-input.asm

```
; Test case generated by KLC3
.ORIG x4000
.FILL x0000           ; INPUT_NUM
.STRINGZ "AAAAAAA"   ; INPUT_STR
.END
```

test0.lcs (to be loaded in Ic3sim)

```
file test.obj
file test0/test0-input.obj
reg PC x3000
```

Students Debug on the LC-3 Simulator with Reverse Execution

The screenshot displays the LC-3 Simulator Interface with the following components:

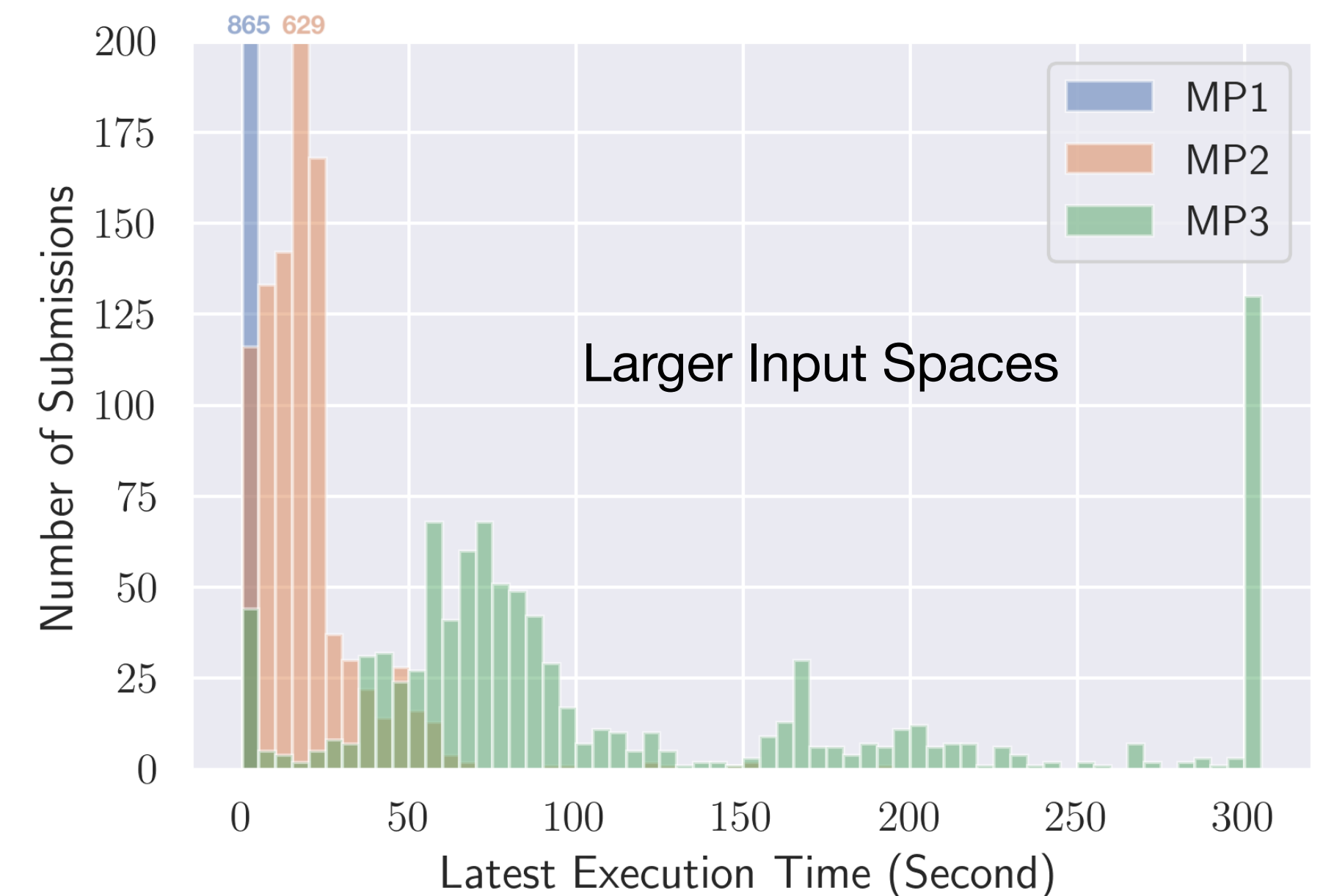
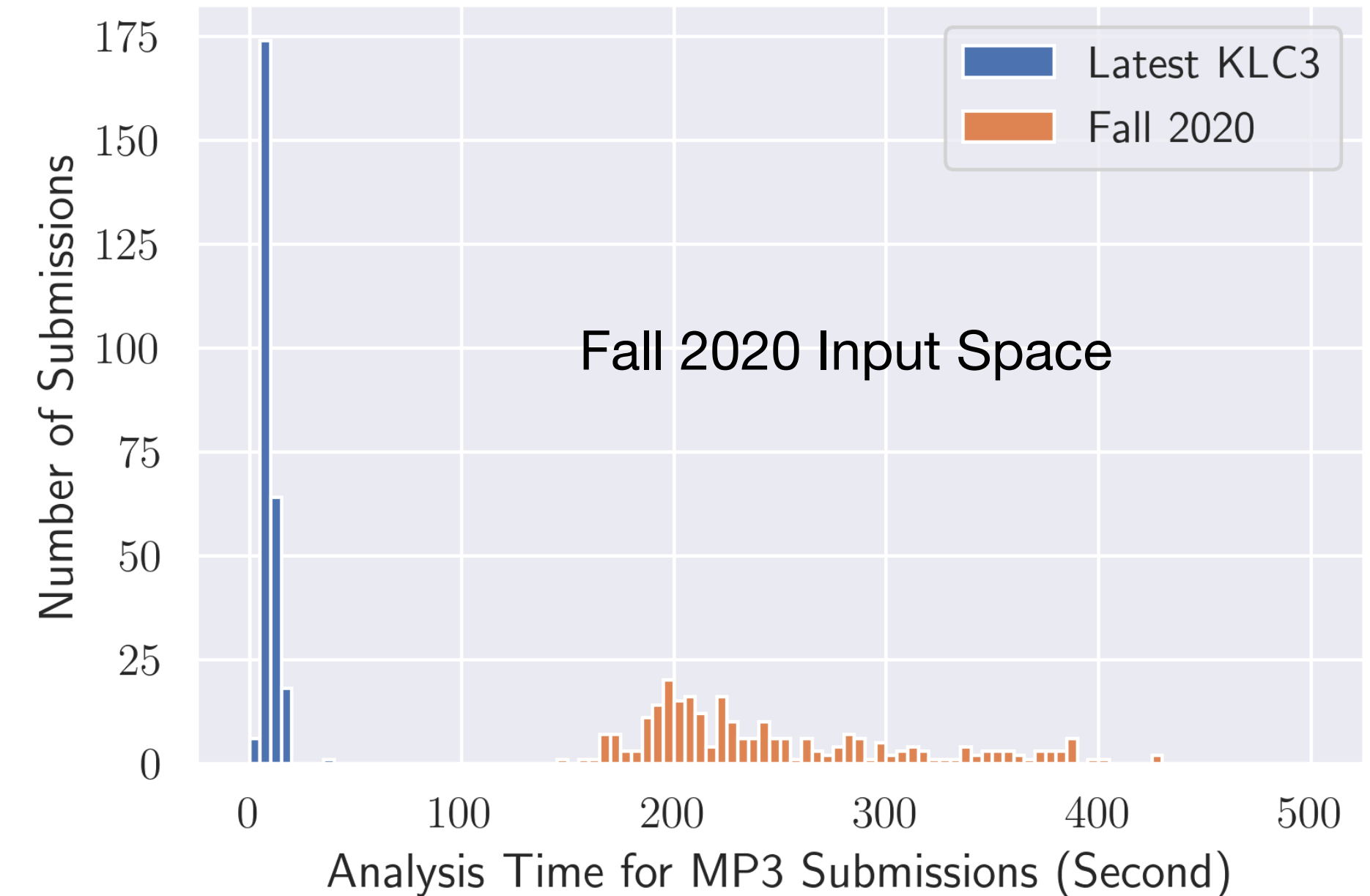
- Registers:** R0 (x0000), R1 (x7FFF), R2 (x0000), R3 (x0000), R4 (x0000), R5 (x0000), R6 (x0000), R7 (x0490), PC (x0494), IR (xB1AE), PSR (x0400), CC (ZERO).
- Memory:** Address x000D, Value x0495.
- Control Buttons:** Next, Step, Finish, Continue, Stop, Clear All Breakpoints, Update Registers.
- Reverse Execution Buttons:** rNext, rStep, rFinish, rContinue. A red arrow points to the rNext button.
- Code Window:** A blue window showing assembly code for TRAP_PUTSP_MSBU and TRAP_PUTSP_DONE.

```
TRAP_PUTSP_MSBU x0480 x1482 ADD R2,R2,R2
x0481 x16FF ADD R3,R3,#-1
x0482 x03F9 BRP TRAP_PUTSP_S_LOOP
x0483 x1020 ADD R0,R0,#0
x0484 x0403 BRZ TRAP_PUTSP_DONE
x0485 xF021 OUT
x0486 x1261 ADD R1,R1,#1
x0487 x0FED BRNZP TRAP_PUTSP_LOOP
TRAP_PUTSP_DONE x0488 x21BE LD R0,OS_R0
x0489 x23BE LD R1,OS_R1
x048A x25BE LD R2,OS_R2
```


Results and Conclusions

- Fall 2020 semester
 - MP3: median 700 lines of code
 - used overly-restricted input space
 - >80% of the students gave positive feedback
- Significant optimization since Fall 2020
- With larger input space
 - 4.08% of samples time out (5 min)
 - all such samples reported issues

**Good application of KLEE:
teach people how to program!**



Thanks

- Questions?



ZJU-UIUC Institute

Zhejiang University / University of Illinois at Urbana-Champaign Institute

